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 the true motion direction between the two time steps is

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

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 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.

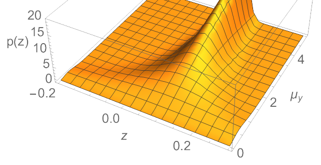
It is reasonable to assume the noise of detection along the u-axis and v-axis are independent so $\rho = 0$. And when representing the center position in pixel, it is also moderate to assume $\sigma_w = \sigma_y = 1$ (also for the ease of presentation). Then, with different true value of $\mu_z = \frac{\mu_y}{\mu_w}$, the visualizations of $p(z)$ over z and μ_y are shown in Figure 5. 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 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.

B. Interpolation by Gaussian Progress Regression

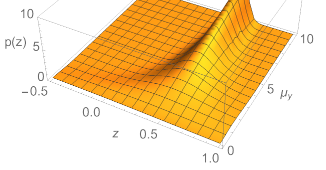
Interpolation as post-processing. Although we focus on developing an online tracking algorithm, we are also interested in whether post-process can further optimize the tracking results in diverse conditions. Despite the failure of GPR in online tracking in Table 6, we continue to study if GPR is better 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.

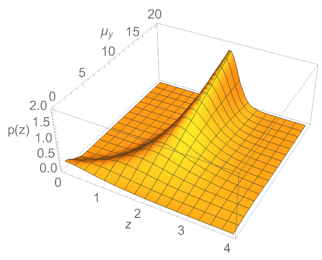
One way is to extend from KF to non-linear filters, such as EKF [30, 52] and UKF [28]. However, for real-world online tracking, they can be hard to be adopted as they



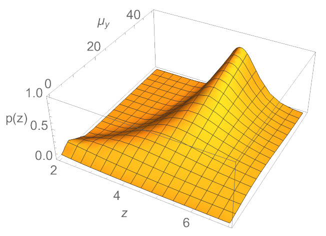
(a) $\mu_z = 0.1$



(b) $\mu_z = 0.5$



(c) $\mu_z = 2$



(d) $\mu_z = 5$

Figure 5. 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

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) [32, 47, 62]: 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

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

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. Until now, we have shown the primary study of using Gaussian Process Regression (GPR) in the online generation of the virtual trajectory in ORU and offline interpolation. But neither of them successfully boosts the tracking performance. Now, We continue to 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 [15]. 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 [15]. 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.

B.2. GPR for Offline Interpolation

We have presented the use of GPR in online virtual trajectory fitting and offline interpolation 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 sets 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.

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**

Table 10. Results on CroHD Head Tracking dataset [56]. Our method uses the detections from HeadHunter [56] or FairMOT [71] to generate new tracks.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP(10^4)↓	FN(10^4)↓	IDs↓	Frag↓
HeadHunter [56]	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 [71]	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
OC-SORT	55.1	80.3	38.0	89.4	54.2
OC-SORT (MOT17)	48.6	71.0	33.3	84.2	51.5

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. On the other hand, 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 the performance by 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 [56]. When considering tracking in the crowd, focusing on only a part of the object can be beneficial [6] as it usually suffers less from occlusion than the full body. This line of study is conducted over hand tracking [40, 50], human pose [66] and head tracking [2, 43, 56] for a while. Moreover, with the knowledge of more fine-grained part trajectory, it can be useful in downstream tasks, such as action recognition [16, 17] and forecasting [7, 9, 31, 33]. 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 [56]. 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 [71] and HeadHunter [56] are extracted from their tracking results downloaded from the official leaderboard [2]. We use the same parameters for OC-SORT as on the other datasets. The results suggest a significant tracking performance improvement compared with the previous methods [56, 71] for human body part tracking. But the tracking performance is still relatively low (HOTA=$\sim$ 40). It is highly related to the difficulty of having accurate detection of tiny objects. Some samples from the test set of HeadTrack are shown in the first two rows of Figure 6.

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

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 [73]	-	61.5	59.6	1.41	20.1	2,583	-	-	-
QDTrack [42]	-	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 [67]	51.4	68.8	61.4	2.29	**14.9**	4,102	8,468	47.7	52.8
TrackFormer [39]	-	**62.5**	60.7	3.28	17.5	2,540	-	-	-
OC-SORT	52.4	58.2	65.1	**0.44**	23.0	**784**	2,006	**57.6**	63.5
OC-SORT + 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 [4]	46.8	57.6	59.1	17.0	20.1	1,210	1,420	47.3	52.7
TransCt [67]	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 [53]	48.9	60.1	61.2	3.80	16.6	2,342	4,320	48.4	52.9
LPC_MOT [13]	49.0	56.3	62.5	1.17	21.3	1,562	1,865	52.4	54.7
OC-SORT	54.3	59.9	67.0	**0.44**	20.2	554	2,345	59.5	65.1
OC-SORT + LI	**55.2**	**61.7**	**67.9**	0.57	19.2	**508**	**805**	**59.8**	**65.9**

Public Tracking on MOT17 and MOT20. Although we use the same object detectors as some selected baselines, there is still variances in detections when compared with other methods. 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 6.

D. Pseudo-code of OC-SORT

See the pseudo-code of OC-SORT in Algorithm. 1.

E. More Results on DanceTrack

To gain more intuition about the improvement of OC-SORT over SORT, we provide more comparisons. In Figure 8, 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 9, we show more samples of trajectory visualizations from SORT and OC-SORT on DanceTrack-val set.

DanceTrack [54] is proposed to encourage better association algorithms instead of carefully tuning detectors. We train YOLOX [19] detector on MOT17 training set only to provide detections on DanceTrack. We find the tracking performance of OC-SORT is already higher than the baselines (Table 11). We believe the potential to improve multi-object tracking by better association strategy is still promising and DanceTrack is a good platform for the evaluation.

F. Integrate Appearance into OC-SORT

OC-SORT is pure motion-based but flexible to integrate with other association cues, such as object appearance. We make an attempt of adding appearance information into OC-SORT and achieve significant performance improvements, validated by experiments on MOT17, MOT20, and DanceTrack. Please refer to Deep OC-SORT [38] for details.

G. More Discussion of State Noise Sensitivity

In Section 3.2.1, we show that the noise of state estimate will be amplified to the noise of velocity estimate. This is because the velocity estimate is correlated to the state estimate. But the analysis is in a simplified model in which velocity itself does not gain noise from the transition directly and the noise of state estimate is i.i.d on different steps. However, in the general case, such a simplification does not hold. We now provide a more general analysis of the state noise sensitivity of SORT.

For the process in Eq 1, we follow the most commonly adapted implementation of Kalman filter [3] and SORT [4] for video multi-object tracking. Instead of writing the mean state estimate, we consider the noisy prediction of state estimate now, which is formulated as

$$\mathbf{x}_{t|t-1} = \mathbf{F}_t \mathbf{x}_{t|t-1} + \mathbf{w}_t, \tag{15}$$

where $\mathbf{w}_t$ is the process noise, drawn from a zero mean multivariate normal distribution, $\mathcal{N}$, with covariance, $\mathbf{w}_t \sim \mathcal{N}(0, \mathbf{Q}_t)$. As $\mathbf{x}_t$ is a seven-tuple, *i.e.* $\mathbf{x}_t = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^\top$, the process noise applies to not just the state estimate but also the velocity estimates. Therefore, for a general form of analysis of temporal error magnification in Eq 5, we would get a different result because not just the position terms but also the velocity terms gain noise from the transition process. And the noise of velocity terms will amplify the noise of position estimate by the transition at

[3]https://github.com/rlabbe/filterpy

[4]https://github.com/abewley/sort

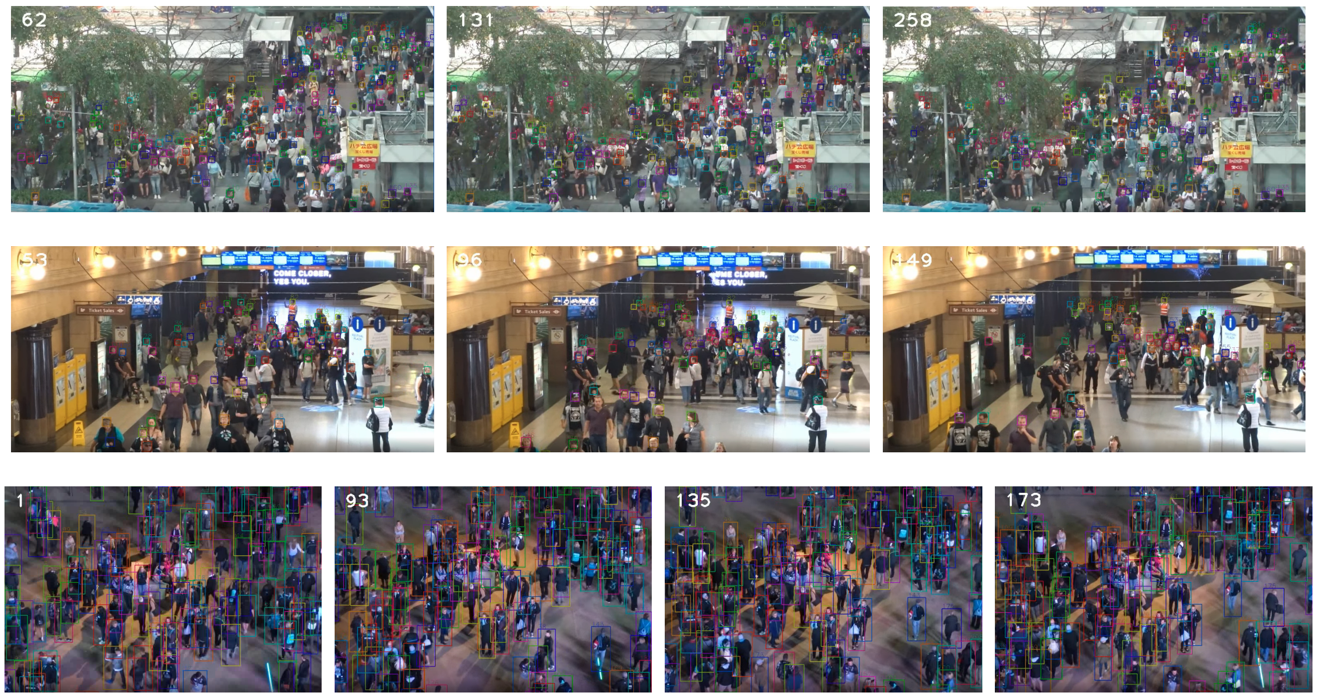


Figure 6. The visualization of the output of OC-SORT on randomly selected samples from the test set of HeadTrack [56] (the first two rows) and MOT20 [14] (the bottom row). These two datasets are both challenging because of the crowded scenes where pedestrians have heavy occlusion with each other. OC-SORT achieves superior performance on both datasets.

the next step. We note the process noise as in practice:

$$\mathbf{Q}_t = \begin{bmatrix} \sigma_u^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_v^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_s^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_r^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\dot{u}}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{\dot{v}}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{\dot{s}}^2 \end{bmatrix}, \tag{16}$$

and the linear transition model as

$$\mathbf{F}_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \tag{17}$$

We assume the time step when a track gets untracked is t_1 and don't consider the noise from previous steps. For simplicity, we assume the motion in the x-direction and y-direction do not correlate. We take the motion on the x-direction as an example without loss of generality:

$$\delta_{u_{t_0}} \sim \mathcal{N}(0, \sigma_u^2), \quad \delta_{\dot{u}_{t_0}} \sim \mathcal{N}(0, \sigma_{\dot{u}}{}^2). \tag{18}$$

On the next step, with no correction from the observation, the error would be accumulated ($\Delta t = 1$),

$$\delta_{u_{t_0+1}} \sim \mathcal{N}(0, 2\sigma_u^2 + \sigma_{\dot{u}}{}^2), \quad \delta_{\dot{u}_{t_0+1}} \sim \mathcal{N}(0, 2\sigma_{\dot{u}}{}^2). \tag{19}$$

Therefore, the accumulation is even faster than we analyze in Section 3.2 as

$$\delta_{u_{t_0+T}} \sim \mathcal{N}(0, (T+1)\sigma_u^2 + \frac{1}{2}T(T+1)\sigma_{\dot{u}}^2). \tag{20}$$

In the practice of SORT, we have to suppress the noise from velocity terms because it is too sensitive. We achieve it by setting a proper value for the process noise $\mathbf{Q}_t$. For example, the most commonly adopted value [5] of $\mathbf{Q}_t$ in SORT is

$$\mathbf{Q}_t = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.01 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0.01 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0.0001 \end{bmatrix}. \tag{21}$$

In such a parameter setting, we have the ratio between

[5]https://github.com/abewley/sort/blob/master/sort.py#L111

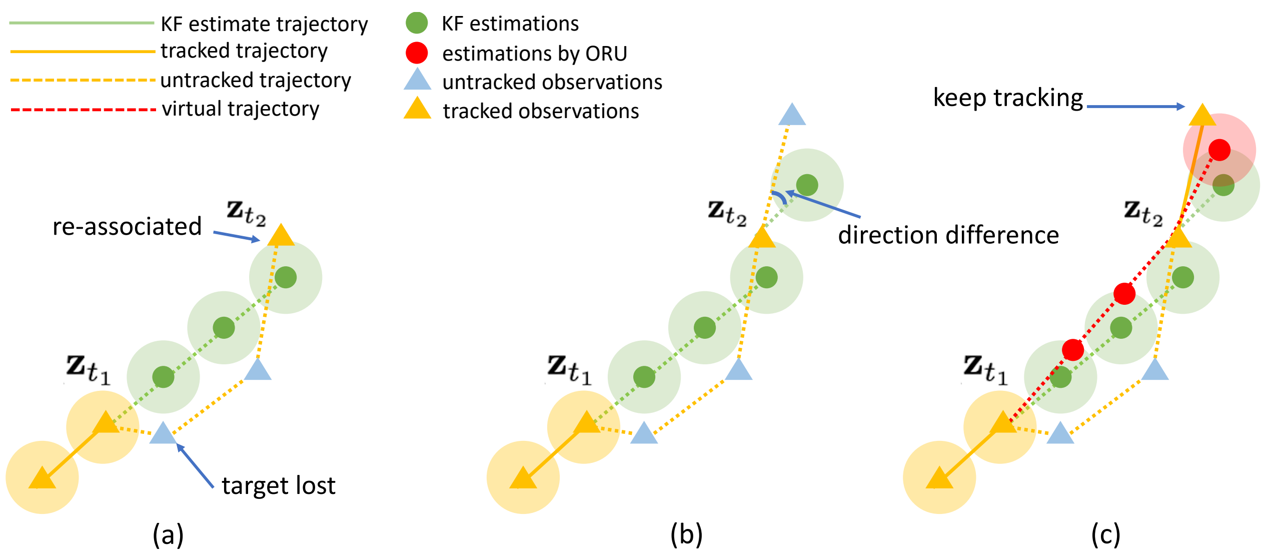


Figure 7. Illustration of how ORU changes the behaviors of SORT after an untracked track is re-associated to an observation. The circle area with shadow indicates the range that an estimate can be associated with observations close enough to it. **(a).** The track is re-associates with an observation $\mathbf{z}_{t_2}$ at the step t_2 after being untracked since the time step t_1. **(b).** Without ORU, on the next step of re-association, even though the KF state is updated by $\mathbf{z}_{t_2}$, there is still a direction difference between the true object trajectory and the KF estimates. Therefore, the track is unmatched with detections again (in blue). **(c).** With ORU, we get a more significant change in the state, especially the motion direction by updating velocity. Now, the state estimate (in red) is closer to the state observation and they can be associated again.

the noise from position terms and velocity terms as

$$\beta = \frac{(T+1)\sigma_u^2}{0.5T(T+1)\sigma_{\dot{u}}^2} = \frac{200}{T}. \tag{22}$$

In practice, a track is typically deleted if it keeps untracked for T_{del} time steps. Usually we set $T_{\text{del}} < 10$, so we have $\beta > 20$. Therefore, we usually consider the noise from velocity terms as secondary. Such a convention allows us to use the simplified model in Section 3.2.1 for noise analysis. But it also brings a side-effect that SORT can't allow the velocity direction of a track to change quickly in a short time interval. We will see later (Section H) that it makes trouble to SORT when non-linear motion and occlusion come together and motivates the design of ORU in OC-SORT.

H. Intuition behind ORU

ORU is designed to fix the error accumulated during occlusion when an untracked track is re-associated with an observation. But in general, the bias in the state estimate $\hat{\mathbf{x}}$ after being untracked for T time steps can be fixed by the update stage once it gets re-associated with an observation. To be precise, the Optimal Kalman gain, *i.e.* $\mathbf{K}_t$, can use the re-associated observation to update the KF posteriori parameters. In general, such an expectation of KF's behavior is reasonable. But because we usually set the corresponding covariance for velocity terms very small (Eq 21), it is difficult for SORT to steer to the correct velocity direction at the step of re-association.

Motivated by such observations, we design ORU. In the simplified model shown in Figure 7, the circle area with the shadow around each estimate footage is the eligible range to associate with observations inside. ORU is designed for the case that a track is re-associated after being untracked. Therefore, the typical situation is as shown in the figure that the true trajectory first goes away from the linear trajectory of KF estimates and then goes closer to it so that there can be a re-association. After the re-association, there would be a cross of the two trajectories.

In SORT, after re-associating with an observation, the direction of the velocity of the previously untracked track still has a significant difference from the true value. This is shown in Figure 7(b). This makes the estimate on the future steps lost again (the blue triangle). The reason is the convention of $\mathbf{Q}$ discussed in Appendix G. Therefore, even though the canonical KF can support fixing the accumulated error during being untracked theoretically, it is very rare in practice. In ORU, we follow the virtual trajectory where we have multiple virtual observations. In this way, even if the value of $\mathbf{Q}[4:,4:]$ is small, we can still have a better-calibrated velocity direction after the time step t_2. We would like to note that the intuition behind ORU is from our observations in practice and based on the common convention of using Kalman filter for multi-object tracking. It does not make fundamental changes to upgrade the power of the canonical Kalman filter.

Here we provide a more formal mathematical expression to compare the behaviors of SORT and OC-SORT. Assume that the track was lost at the time step t_1 and re-associated at t_2. We assume the mean state estimate is

$$\hat{\mathbf{x}}_{t_1|t_1} = [u_1, v_1, s_1, r_1, \dot{u}_1, \dot{v}_1, \dot{s}_1]^\top, \tag{23}$$

and the covariance at t_1 is

$$\mathbf{P}_{t_1|t_1} = \begin{bmatrix} \sigma_{u_1}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{v_1}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{s_1}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{r_1}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\dot{u}_1}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{\dot{v}_1}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{\dot{s}_1}^2 \end{bmatrix}. \tag{24}$$

Then, because the covariance expands from the input of process noise at each step of *predict*, at t_2, we have the priori estimates ($t_\Delta = t_2 - t_1$) of state

$$\hat{\mathbf{x}}_{t_2|t_2-1} = [u_2, v_2, s_2, r_2, \dot{u}_2, \dot{v}_2, \dot{s}_2]^\top, \tag{25}$$

with

$$\begin{aligned} u_2 &= u_1 + \dot{u}_1 t_\Delta, \\ v_2 &= v_1 + \dot{v}_1 t_\Delta, \\ s_2 &= s_1 + \dot{s}_1 t_\Delta, \\ r_2 &= r_1, \\ \dot{u}_2 &= \dot{u}_1, \\ \dot{v}_2 &= \dot{v}_1, \\ \dot{s}_2 &= \dot{s}_1. \end{aligned} \tag{26}$$

And the priori covariance

$$\mathbf{P}_{t_2|t_2-1} = \begin{bmatrix} \sigma_{u_2}^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & \sigma_{v_2}^2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_{s_2}^2 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \sigma_{r_2}^2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \sigma_{\dot{u}_2}^2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & \sigma_{\dot{v}_2}^2 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & \sigma_{\dot{s}_2}^2 \end{bmatrix}, \tag{27}$$

with

$$\begin{aligned} \sigma_{u_2}^2 &= \sigma_{u_1}^2 + t_\Delta(\sigma_u^2 + \sigma_{\dot{u}_1}^2) + \frac{t_\Delta(t_\Delta - 1)}{2}\sigma_{\dot{u}}^2, \\ \sigma_{v_2}^2 &= \sigma_{v_1}^2 + t_\Delta(\sigma_v^2 + \sigma_{\dot{v}_1}^2) + \frac{t_\Delta(t_\Delta - 1)}{2}\sigma_{\dot{v}}^2, \\ \sigma_{s_2}^2 &= \sigma_{s_1}^2 + t_\Delta(\sigma_s^2 + \sigma_{\dot{s}_1}^2) + \frac{t_\Delta(t_\Delta - 1)}{2}\sigma_{\dot{s}}^2, \\ \sigma_{r_2}^2 &= \sigma_{r_1}^2 + t_\Delta \sigma_r^2, \\ \sigma_{\dot{u}_2}^2 &= \sigma_{\dot{u}_1}^2 + t_\Delta \sigma_{\dot{u}}^2, \\ \sigma_{\dot{v}_2}^2 &= \sigma_{\dot{v}_1}^2 + t_\Delta \sigma_{\dot{v}}^2, \\ \sigma_{\dot{s}_2}^2 &= \sigma_{\dot{s}_1}^2 + t_\Delta \sigma_{\dot{s}}^2. \end{aligned} \tag{28}$$

Now, SORT will keep going forward as normal. Therefore, with the re-associated observation $\mathbf{z}_{t_2}$, we have

$$\textit{SORT} \begin{cases} \hat{\mathbf{x}}_{t_2|t_2} = \hat{\mathbf{x}}_{t_2|t_2-1} + \mathbf{K}_{t_2}(\mathbf{z}_{t_2} - \mathbf{H}\hat{\mathbf{x}}_{t_2|t_2-1}), \\ \mathbf{P}_{t_2|t_2} = (\mathbf{I} - \mathbf{K}_{t_2}\mathbf{H})\mathbf{P}_{t_2|t_2-1} \end{cases} \tag{29}$$

where the observation model is

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}, \tag{30}$$

and the Kalman gain is

$$\mathbf{K}_{t_2} = \mathbf{P}_{t_2|t_2-1}\mathbf{H}^\top(\mathbf{H}\mathbf{P}_{t_2|t_2-1}\mathbf{H}^\top + \mathbf{R}_{t_2})^{-1}. \tag{31}$$

On the other hand, OC-SORT will replay Kalman filter *predict* on a generated virtual trajectory to gain the posteriori estimates on t_2 (ORU). With the default linear motion analysis, we have the virtual trajectory as

$$\tilde{\mathbf{z}}_t = \mathbf{z}_{t_1} + \frac{t - t_1}{t_2 - t_1}(\mathbf{z}_{t_2} - \mathbf{z}_{t_1}), t_1 < t < t_2. \tag{32}$$

Now, to derive the posteriori estimate, we will run the loop between *predict* and *re-update* from t_1 to t_2.

$$\textit{OC-SORT} \begin{cases} \hat{\mathbf{x}}_{t|t} = \mathbf{F}\hat{\mathbf{x}}_{t-1|t-1} + \mathbf{K}_t(\tilde{\mathbf{z}}_t - \mathbf{H}\mathbf{F}\hat{\mathbf{x}}_{t-1|t-1}) \\ \mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H})(\mathbf{F}\mathbf{P}_{t-1|t-1}\mathbf{F}^\top + \mathbf{Q}_t) \end{cases} \tag{33}$$

where the Kalman gain is

$$\mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^\top(\mathbf{H}\mathbf{P}_{t|t-1}\mathbf{H}^\top + \mathbf{R}_t)^{-1}, \tag{34}$$

and we can always rewrite it with

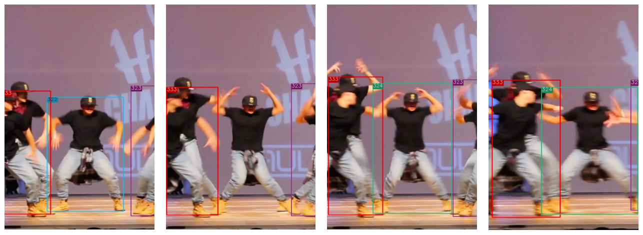
$$\mathbf{P}_{t|t-1} = \mathbf{F}\mathbf{P}_{t-1|t-1}\mathbf{F}^\top + \mathbf{Q}_t. \tag{35}$$

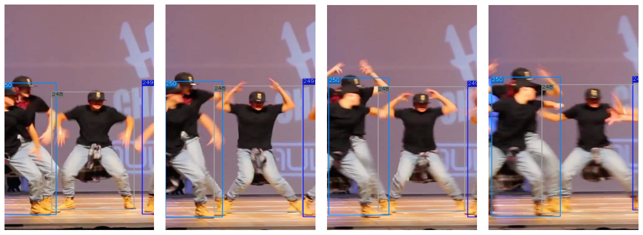
In the common practice of Kalman filter, we assume a constant set of Gaussian noise for the process noise $\mathbf{Q}_t$. This assumption typically can't hold in practice. This makes the conflict that when there are consistent observations over time, we require a small process noise for multi-object tracking in high-frame-rate videos. However, when there is a period of observation missing, the direction difference between the true direction and the direction maintained by the linear motion assumption grows. This causes the failure of SORT to consistently track previously lost targets even after re-association.

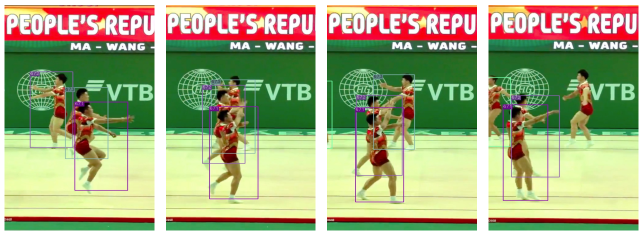
We show the different outcomes of SORT and OC-SORT upon re-associating lost targets in Eq 29 and Eq 33. Analyzing their difference more deeply will require more assumptions of the underlying true object trajectory and the observations. Therefore, instead of theoretical proof, we demonstrate the gain of performance from OC-SORT over SORT empirically as shown in the experiments.

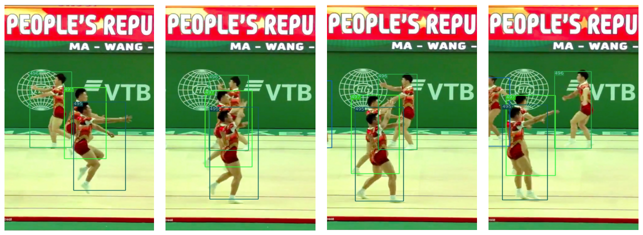
Algorithm 1: Pseudo-code of OCSORT.

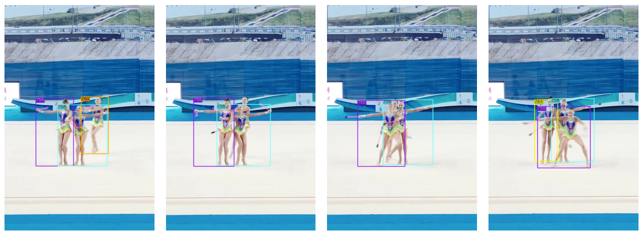
```
Input: Detections $\mathcal{Z} = \{\mathbf{z}_k^i | 1 \leq k \leq T, 1 \leq i \leq N_k\}$; Kalman Filter KF; threshold to remove untracked tracks $t_{\text{expire}}$
Output: The set of tracks $\mathcal{T} = \{\tau_i\}$
1  Initialization: $\mathcal{T} \leftarrow \emptyset$ and KF;
2  for timestep $t \leftarrow 1 : T$ 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      $\mathcal{Z} \leftarrow$ Historical observations on the existing tracks
6      $C_t \leftarrow C_{\text{IoU}}(\hat{\mathbf{X}}_t, \mathbf{Z}_t) + \lambda C_v(\mathcal{Z}, \mathbf{Z}_t)$ /* 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 ORU 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: $\mathcal{T}$
```

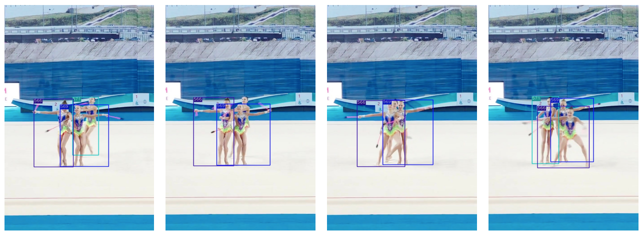

(a) SORT: dancetrack0036

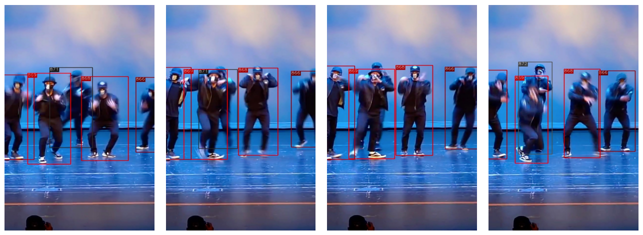

(b) OC-SORT: dancetrack0036

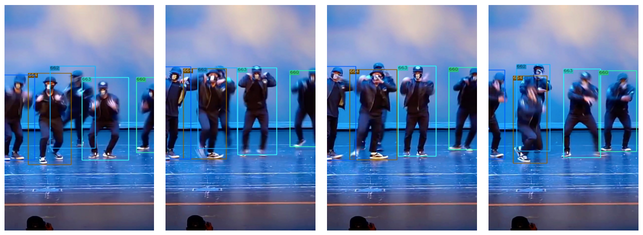

(c) SORT: dancetrack0054

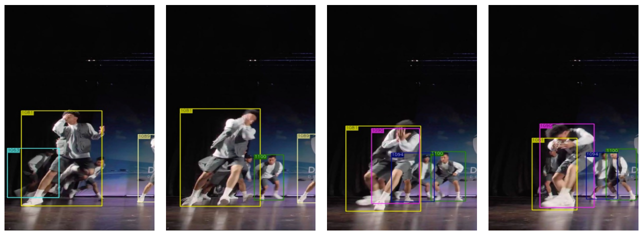

(d) OC-SORT: dancetrack0054

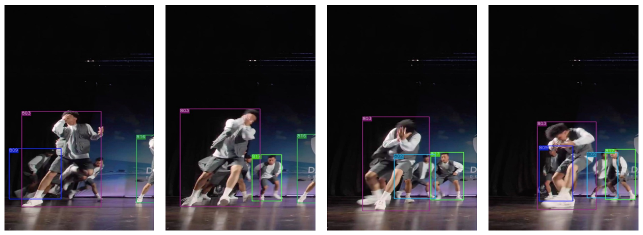

(e) SORT: dancetrack0064

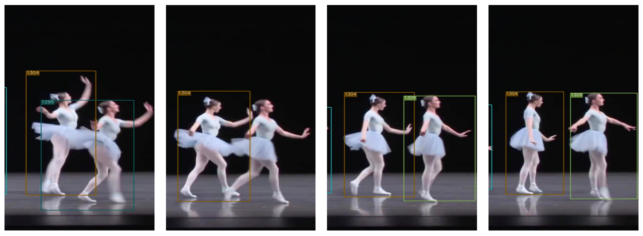

(f) OC-SORT: dancetrack0064


(g) SORT: dancetrack0078


(h) OC-SORT: dancetrack0078


(i) SORT: dancetrack0089


(j) OC-SORT: dancetrack0089


(k) SORT: dancetrack0100

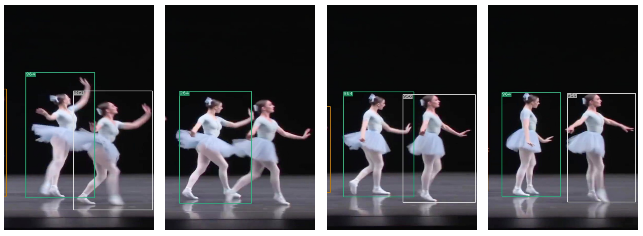
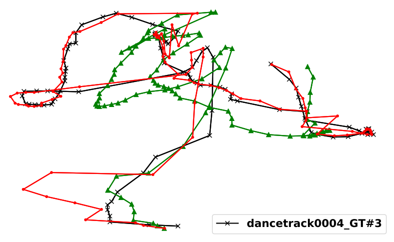
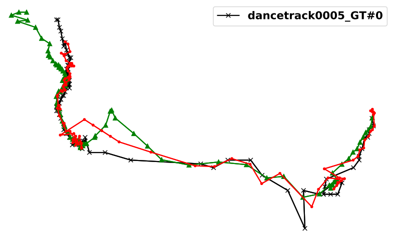

(l) OC-SORT: dancetrack0100

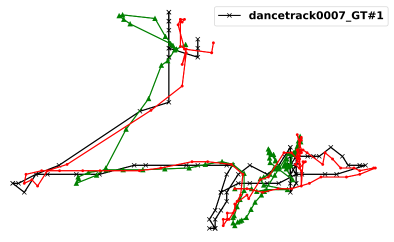
Figure 8. More samples where SORT suffers from the fragmentation and ID switch of tracks from occlusion or non-linear motion but OC-SORT survives. To be precise, the issue happens on the objects by SORT at: (a) #322 $\rightarrow$ #324; (c) ID switch between #672 and #673, later #673 being lost; (e) #760 $\rightarrow$ #761; (g) #871 $\rightarrow$ #872; (i) #1063 $\rightarrow$ #1090, then ID switch with #1081; (l) #1295 $\rightarrow$ #1304. We select samples from diverse scenes, including street dance, classic dance and gymnastics. Best viewed in color and zoomed in.



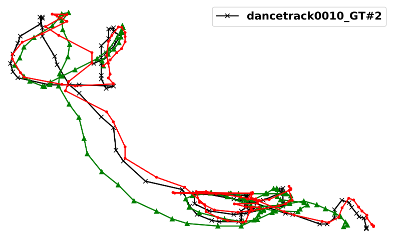
(a) GT #3 on video #0003



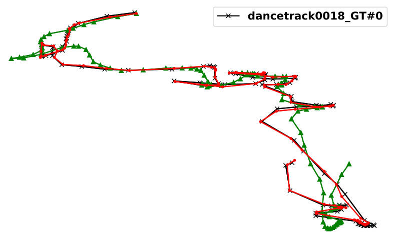
(b) GT #0 on video #0005



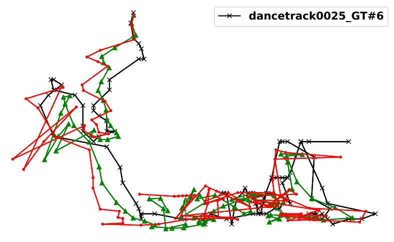
(c) GT #1 on video #0007



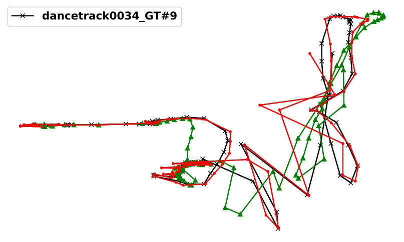
(d) GT #2 on video #0010



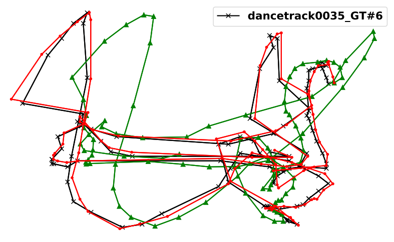
(e) GT #0 on video #0018



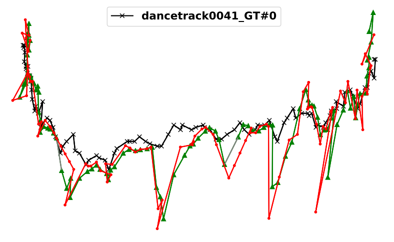
(f) GT #6 on video #0025



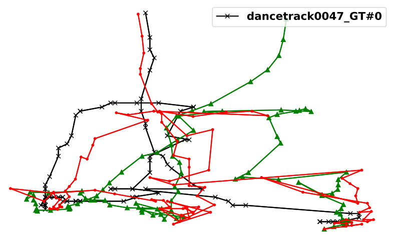
(g) GT #9 on video #0034



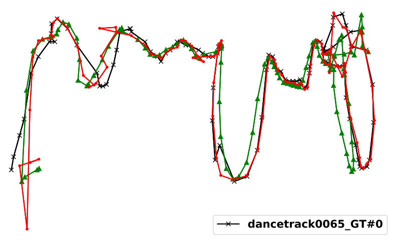
(h) GT #6 on video #0035



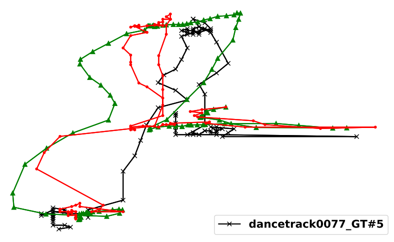
(i) GT #0 on video #0041



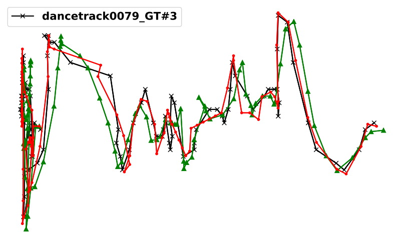
(j) GT #0 on video #0047



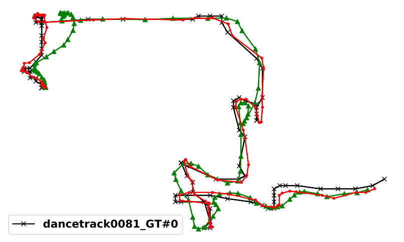
(k) GT #0 on video #0065



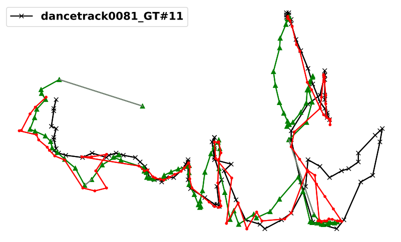
(l) GT #5 on video #0077



(m) GT #3 on video #0079



(n) GT #0 on video #0081



(o) GT #11 on video #0081

Figure 9. Randomly selected object trajectories on the videos from Dancetrack-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.