JUNE 18-22, 2023
CVPR
VANCOUVER, CANADA

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

Jinkun Cao[1], Jiangmiao Pang[2], Xinshuo Weng[3], Rawal Khirodkar[1], Kris Kitani[1]

[1]Carnegie Mellon University [2]Shanghai AI Lab [3]Nvidia



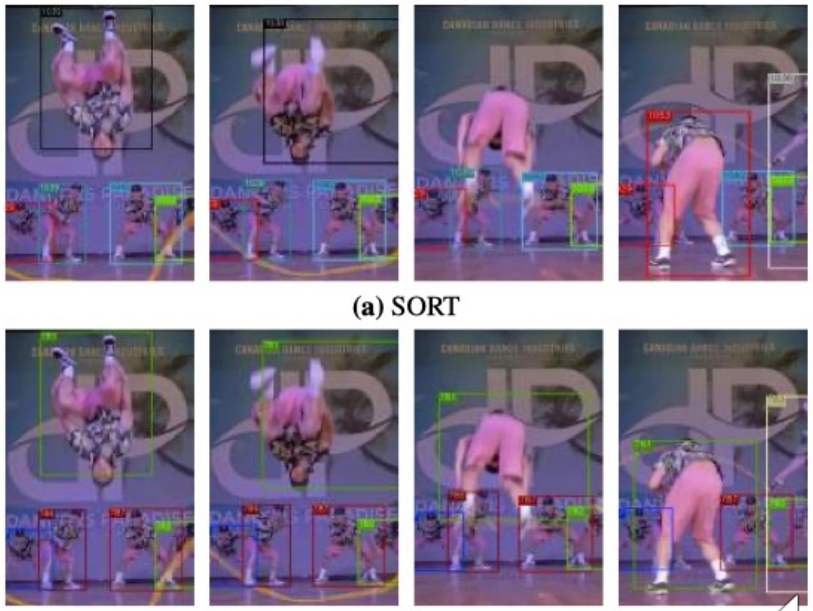


WED-AM-139

Overview

We propose a MOT algorithm by improving SORT's limitations.

1. Simple: no training required, a pure filtering-based method.
2. Effective: state-of-the-art on multiple MOT benchmarks[1].
3. Efficient: Run at >500FPS on a single CPU[2].



(a) SORT

(b) The proposed OC-SORT

1: With YOLO-X as the detector.

2: Tested on KITTI dataset.

Motivation

SORT[1] applies Kalman filter for multi-object tracking under the assumption of linear motion.

- Works well only when video frame rate is high and consistent measurement is provided.
- Fails when measurement is missing during a time interval, when Kalman filter propagates state estimates by linear motion assumption.

[1] Bewley, Alex, et al. "Simple online and realtime tracking." *2016 IEEE international conference on image processing (ICIP)*. IEEE, 2016.

Preliminary: SORT

- Tracking-by-detection: works with an off-the-shelf detector.
- Detector provides an observation/measurement of target states and Kalman filter provides an estimate following linear motion assumption.
- Determines posteriori state estimate by combing the two states.

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

Preliminary: SORT

- Tracking-by-detection: works with an off-the-shelf detector.
- Detector provides an observation/measurement of target states and Kalman filter provides an estimate following linear motion assumption.
- Determines posteriori state estimate by combing the two states.

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

When observation is missing:

$$\hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1}, \mathbf{P}_{t|t} = \mathbf{P}_{t|t-1}.$$

Why SORT fails

- Sensitive to State Noise: on high-frame-rate videos, the variance of velocity is significant even with tiny position variance.

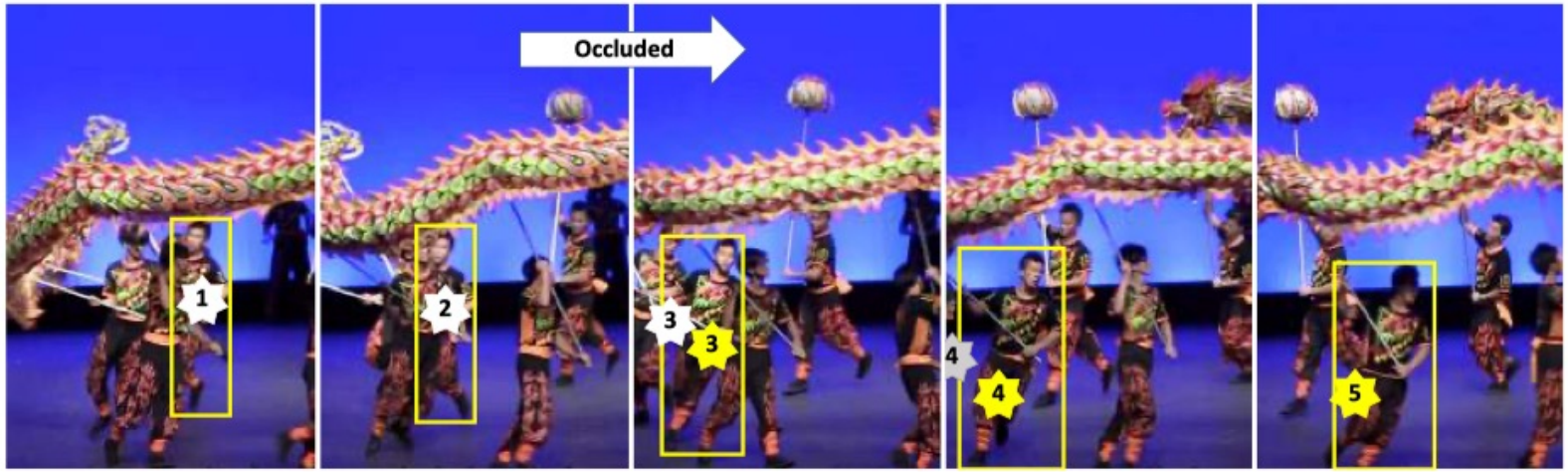
Why SORT fails

- Sensitive to State Noise: on high-frame-rate videos, the variance of velocity is significant even with tiny position variance.
- Temporal Error Magnification: without observation, the state noise can’t be corrected and is accumulated to distort the KF parameters.

Why SORT fails

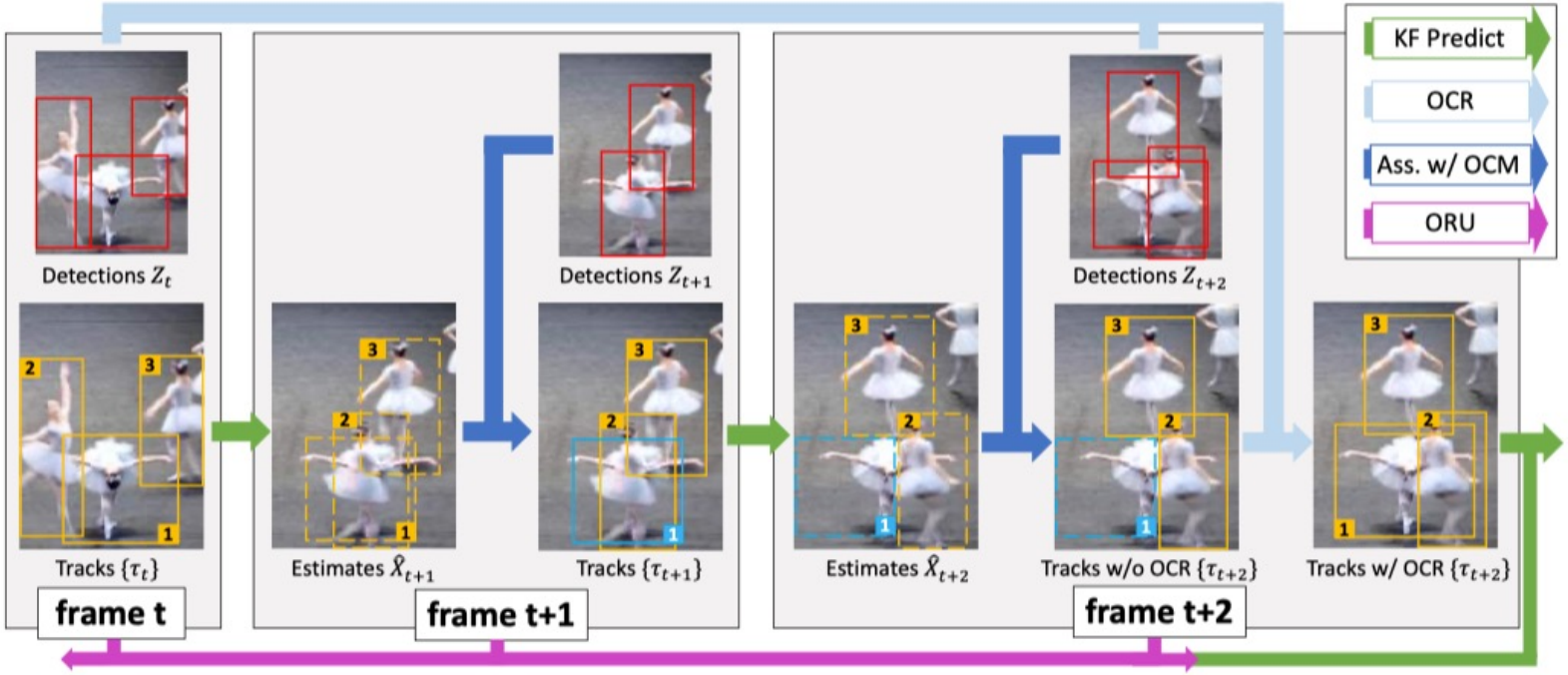
- Sensitive to State Noise: on high-frame-rate videos, the variance of velocity is significant even with tiny position variance.
- Temporal Error Magnification: without observation, the state noise can’t be corrected will is accumulated to distort the KF parameters.
- Being Estimation-Centric: SORT fails as it trusts the priori estimations even when it can’t be trusted anymore.

Proposed method: OC-SORT



Key idea: leverage the observation to avoid the error accumulated in KF parameters when KF propagates without observations.

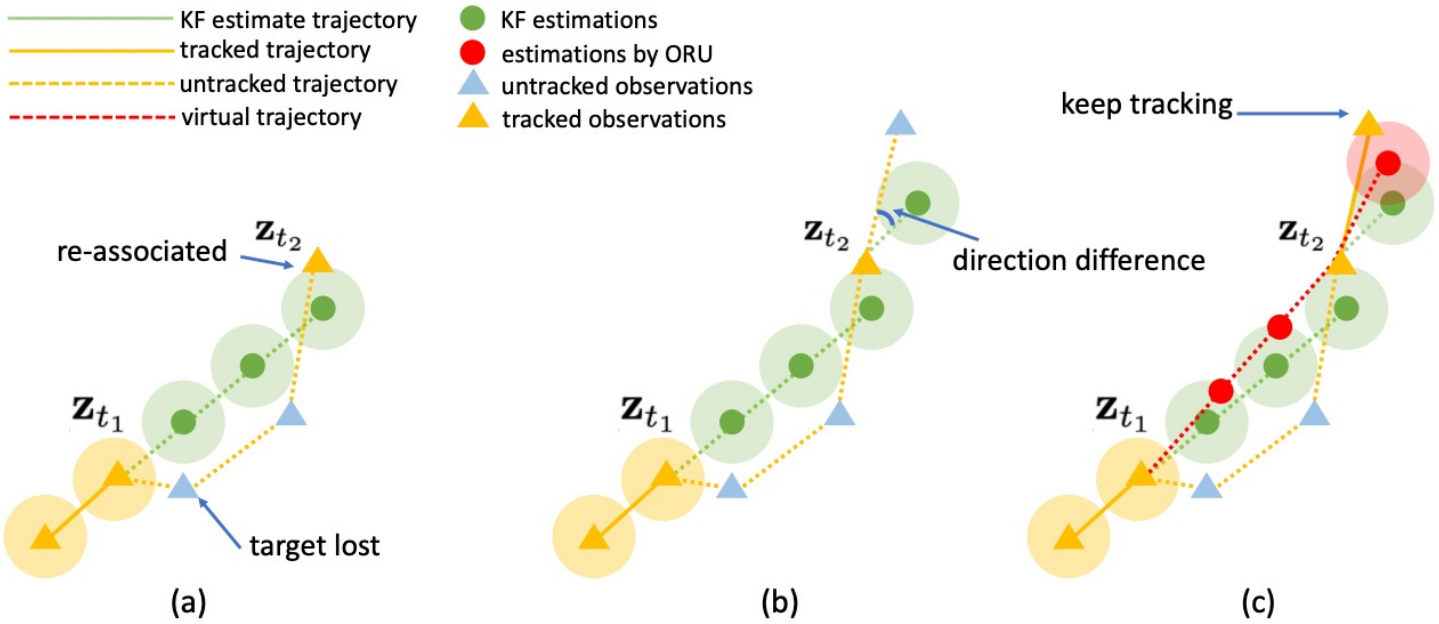
OC-SORT: Architecture



Two main components upon SORT:

1. Observation-centric Re-Update (ORU)
2. Observation-Centric Momentum (OCM)

Observation-centric Re-Update (ORU)



After re-association with an observation, re-update KF parameters:

$$\tilde{\mathbf{z}}_t = Traj_{\text{virtual}}(\mathbf{z}_{t_1}, \mathbf{z}_{t_2}, t), t_1 < t < t_2. \quad \textit{re-update} \begin{cases} \mathbf{K}_t = \mathbf{P}_{t|t-1}\mathbf{H}_t^\top(\mathbf{H}_t\mathbf{P}_{t|t-1}\mathbf{H}_t^\top + \mathbf{R}_t)^{-1} \\ \hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t(\tilde{\mathbf{z}}_t - \mathbf{H}_t\hat{\mathbf{x}}_{t|t-1}) \\ \mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t\mathbf{H}_t)\mathbf{P}_{t|t-1} \end{cases}$$

Observation-Centric Momentum (OCM)

Linear motion assumption includes not just linear position sequence but also consistent velocity. OCM adds the velocity consistency into the cost function for association.

given two observations $(\mu_{u_{t_1}}, \mu_{u_{t_1}})$ and $(\mu_{u_{t_1}}, \mu_{u_{t_1}})$

$$\theta = \arctan\left(\frac{\mu_{v_{t_1}} - \mu_{v_{t_2}}}{\mu_{u_{t_1}} - \mu_{u_{t_2}}}\right)$$

Observation-Centric Momentum (OCM)

Linear motion assumption includes not just linear position sequence but also consistent velocity. OCM adds the velocity consistency into the cost function for association.

given two observations $(\mu_{u_{t_1}}, \mu_{u_{t_1}})$ and $(\mu_{u_{t_1}}, \mu_{u_{t_1}})$

$$\theta = \arctan\left(\frac{\mu_{v_{t_1}} - \mu_{v_{t_2}}}{\mu_{u_{t_1}} - \mu_{u_{t_2}}}\right)$$

$C_v = \Delta\theta$ where $\Delta\theta = |\, \theta^{track} - \theta^{intention} \,|$

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

Experiments

Table 1. Results on MOT17-test with the private detections. ByteTrack and OC-SORT share detections.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP(10^4)↓	FN(10^4)↓	IDs↓	Frag↓	AssA↑	AssR↑
FairMOT [71]	59.3	73.7	72.3	2.75	11.7	3,303	8,073	58.0	63.6
TransCt [67]	54.5	73.2	62.2	2.31	12.4	4,614	9,519	49.7	54.2
TransTrk [55]	54.1	75.2	63.5	5.02	8.64	3,603	4,872	47.9	57.1
GRTU [60]	62.0	74.9	75.0	3.20	10.8	**1,812**	**1,824**	62.1	65.8
QDTrack [42]	53.9	68.7	66.3	2.66	14.7	3,378	8,091	52.7	57.2
MOTR [69]	57.2	71.9	68.4	2.11	13.6	2,115	3,897	55.8	59.2
PermaTr [57]	55.5	73.8	68.9	2.90	11.5	3,699	6,132	53.1	59.8
TransMOT [12]	61.7	76.7	75.1	3.62	9.32	2,346	7,719	59.9	66.5
GTR [75]	59.1	75.3	71.5	2.68	11.0	2,859	-	61.6	-
DST-Tracker [8]	60.1	75.2	72.3	2.42	11.0	2,729	-	62.1	-
MeMOT [5]	56.9	72.5	69.0	2.72	11.5	2,724	-	55.2	-
UniCorn [68]	61.7	77.2	75.5	5.01	**7.33**	5,379	-	-	-
ByteTrack [70]	63.1	**80.3**	77.3	2.55	8.37	2,196	2,277	62.0	**68.2**
OC-SORT	**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 OC-SORT share detections.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP(10^4)↓	FN(10^4)↓	IDs↓	Frag↓	AssA↑	AssR↑
FairMOT [71]	54.6	61.8	67.3	10.3	8.89	5,243	7,874	54.7	60.7
TransCt [67]	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 [61]	53.6	67.1	67.5	3.19	13.5	3,131	9,875	52.7	58.5
TransMOT [12]	61.9	77.5	75.2	3.42	**8.08**	1,615	2,421	60.1	66.3
MeMOT [5]	54.1	63.7	66.1	4.79	13.8	1,938	-	55.0	-
ByteTrack [70]	61.3	**77.8**	75.2	2.62	8.76	1,223	1,460	59.6	66.2
OC-SORT	**62.1**	75.5	**75.9**	**1.80**	10.8	**913**	**1,198**	**62.0**	**67.5**

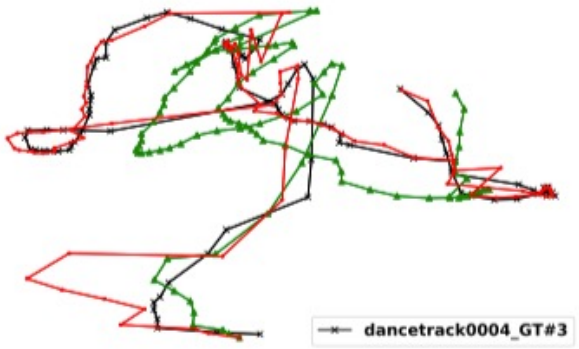
Table 3. Results on DanceTrack test set. Methods in the blue block share the same detections.

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

Table 4. Results on KITTI-test. Our method uses the same detections as PermaTr [57]

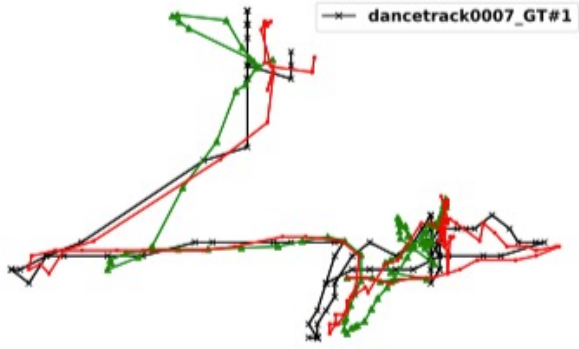
	Car					Pedestrian				
Tracker	HOTA↑	MOTA↑	AssA↑	IDs↓	Frag↓	HOTA↑	MOTA↑	AssA↑	IDs↓	Frag↓
IMMDP [65]	68.66	82.75	69.76	211	181	-	-	-	-	-
SMAT [21]	71.88	83.64	72.13	**198**	294	-	-	-	-	-
TrackMPNN [45]	72.30	87.33	70.63	481	237	39.40	52.10	35.45	626	669
MPNTrack [4]	-	-	-	-	-	45.26	46.23	47.28	397	1,078
CenterTr [73]	73.02	88.83	71.18	254	227	40.35	53.84	36.93	425	618
LGM [59]	73.14	87.60	72.31	448	**164**	-	-	-	-	-
TuSimple [11]	71.55	86.31	71.11	292	218	45.88	57.61	47.62	246	651
PermaTr [57]	**77.42**	**90.85**	**77.66**	275	271	47.43	65.05	43.66	483	703
OC-SORT	74.64	87.81	74.52	257	318	52.95	62.00	57.81	**181**	**598**
OC-SORT + HP	76.54	90.28	76.39	250	280	**54.69**	**65.14**	**59.08**	184	609

OC-SORT v.s. SORT



(a) GT #3 on video #0003

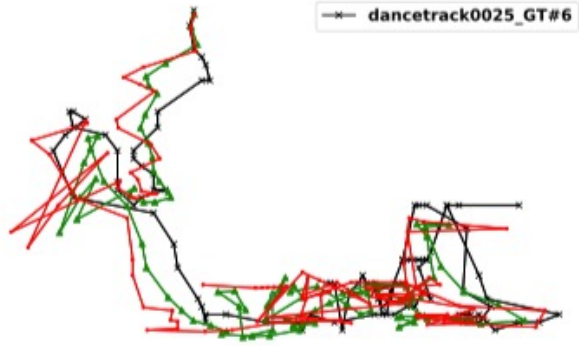
(b) GT #0 on video #0005



(c) GT #1 on video #0007

(d) GT #2 on video #0010

(e) GT #0 on video #0018



(f) GT #6 on video #0025

Demo

https://www.youtube.com/watch?v=qv6gl4h0dvg

Paper & Code

Full paper

Github

mmtracking version