Supplementary Materials for Omnidirectional Multi-Object Tracking

Kai Luo[1,*] Hao Shi[2,*] Sheng Wu[1] Fei Teng[1] Mengfei Duan[1] Chang Huang[1]
Yuhang Wang[1] Kaiwei Wang[2] Kailun Yang[1,†]
[1]Hunan University [2]Zhejiang University

1. Annotation of the QuadTrack Dataset

In the annotation process of the established QuadTrack dataset, we used CVAT [4], an open-source annotation tool that supports tasks such as object detection, object tracking, and instance segmentation. CVAT offers both local and online versions, providing high flexibility for users. Prior to annotation, we preprocessed the dataset by selecting representative scenes, including 32 sequences (seq), with 16 sequences allocated for training and 16 for testing. Each sequence contains 600 frames with a frame rate of approximately 10FPS, resulting in a duration of about 60 seconds per sequence. Furthermore, to assist annotators in better semantic understanding and precise labeling, we unfolded the images into a 2048×480 panoramic layout via equirectangular projection. For the bounding boxes at the image borders, we ensured continuous tracking, guaranteeing that the same object in the surrounding environment maintained a unique ID. The minimum bounding box area was set to 800 pixels, and any targets smaller than this area were ignored. The QuadTrack dataset includes two common object classes: *car* and *person*.

Upon completion of the annotation process, the final annotation attributes were thoroughly reviewed and validated through a filtering and cross-validation procedure to ensure data accuracy. After ensuring the correctness of the annotations, the final annotation attributes were formatted into the MOT standard [10]. Example of ground truth:

```
# MOT format
# f_id, t_id, x, y, w, h, conf, cls, vis
# data
1,1,733.67,281.66,34.78,106.81,1,1,1.0
1,2,557.87,268.05,24.36,128.58,1,1,1.0
1,3,382.33,316.41,110.61,61.49,1,2,1.0
1,4,000.00,301.35,35.02,82.89,1,2,1.0
1,5,1917.7,278.79,20.70,97.98,1,1,1.0
...
```

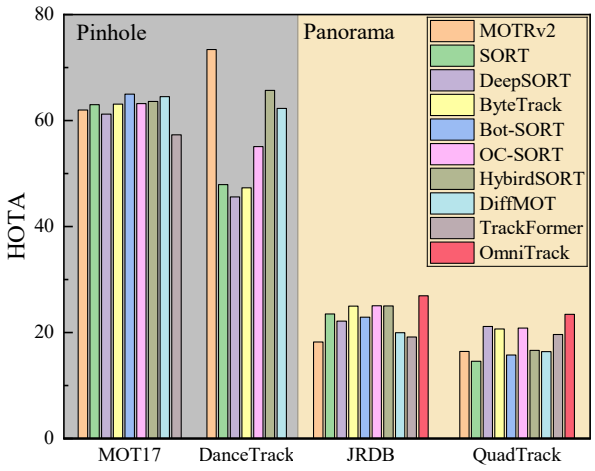

Figure 1. Comparison of state-of-the-art methods on different datasets. **Pinhole** refers to Multi-Object Tracking (MOT) datasets that utilize pinhole camera images, whereas **Panorama** refers to MOT datasets that employ panoramic images.

For a comprehensive description of the attributes in the dataset, please refer to Tab. 1. This annotation format, commonly used in Multi-Object Tracking (MOT) research, provides a structured and standardized method for organizing the data. The inclusion of essential attributes such as object identity, bounding box coordinates, and visibility status is critical for training and assessing tracking models in dynamic, real-world environments. In Fig. 4, examples from the QuadTrack dataset are shown, demonstrating the diversity of scenes and the visual presentation of annotations.

2. Additional Ablation Studies and Analyses

2.1. More Analyses of the DynamicSSM Block

We provide a more detailed discussion on the components of the DynamicSSM Block in Tab. 2. The DynamicSSM Block is composed of three primary operations: (i) distortion alleviation, as described in the main text Equation 9, (ii) addressing lighting and color inconsistencies, as detailed in the main text Equation 10, and (iii) enhancing feature rep-

*Equal contribution
†Correspondence: kailun.yang@hnu.edu.cn

Pos.	Key	Explanation
1	Frame_id	Represents the frame ID.
2	Track_id	A unique identifier for each object. A value of -1 indicates a detection item.
3	Left	Coordinates of the top-left corner of the object bounding box.
4	Top	Coordinates of the top-left corner of the object bounding box.
5	Width	Width of the object bounding box.
6	Height	Height of the object bounding box.
7	Confidence	It acts as a flag whether the entry is to be considered (1) or ignored (0).
8	Class	Indicates the type of object annotated.
9	Visibility	Visibility ratio, a number between 0 and 1 that says how much of that object is visible.

Table 1. Detailed explanation of the annotation attributes for the QuadTrack dataset, including the meaning of each position.

Figure 2. The QuadTrack dataset presents several significant challenges. The images labeled (a), (b), (c), and (d) illustrate continuous frames 80 to 84 from a sequence, with corresponding magnified views shown on the right. In these magnifications, solid rectangular boxes represent the Ground Truth (GT) for the current frame, while dashed boxes correspond to the GT from the preceding frame. One notable challenge is motion blur, particularly evident in the magnified view of frame (b), where the bionic gait introduces substantial blur to the target object. Moreover, there is considerable positional displacement between adjacent frames, as demonstrated in the magnified views of frames (c) and (d). The panoramic images also present inherent exposure issues, displaying both overexposed and underexposed regions, as seen in (a). Finally, the continuity inherent in the panoramic images presents an additional critical factor for the tracking task.

resentation, as formulated in the main text Equation 11. As shown in Tab. 2, all three operations individually contribute to improved performance, and their combination results in the best overall performance. A comparison between experiments ① and ④ demonstrates that integrating all three operations in the DynamicSSM Block leads to an overall HOTA improvement of 1.82%.

2.2. More Analyses of the CircularStatE Module

In the CircularStatE Module, we designed a key component, the DynamicSSM Block, to address challenges such as distortion and lighting inconsistencies inherent in panoramic images. Compared to convolutional networks, the DynamicSSM Block offers a significant performance advantage in handling these issues. To further explore the impact of convolutional networks on multi-scale features, we conducted

Exp.	Dconv	SSM	Fusion	HOTA↑	IDF1↑	OSPA↓
①	-	-	-	23.30	25.50	0.93
②	-	✓	✓	24.82	27.17	**0.92**
③	✓	-	✓	24.81	26.98	**0.92**
④	✓	✓	-	24.72	26.66	**0.92**
④	✓	✓	✓	**25.12**	**27.42**	0.93

Table 2. Ablation of the DynamicSSM Block: Dconv represents deformable convolution (Equation 9 in the main text), SSM denotes the state-space model (Equation 10 in the main text), and Fusion refers to the integration of residual features (Equation 11 in the main text).

Exp.	$\mathcal{S}_5$	$\mathcal{S}_4$	$\mathcal{S}_3$	HOTA↑	IDF1↑	OSPA↓
①	-	-	-	23.296	25.496	0.93415
②	Conv	Conv	Conv	23.565	25.814	**0.90931**
③	Conv	-	-	**24.107**	**26.374**	0.92567
④	-	Conv	-	23.814	26.083	0.92624
⑤	-	-	Conv	23.721	25.565	0.91992

Table 3. Analysis of the impact of convolution in the CircularStatE Module. S_3, S_4, and S_5 represent multi-scale features extracted from the backbone [6]. *Conv* represent convolution.

additional experiments, as summarized in Tab. 3. The results show that applying a convolutional network to the S5 scale yielded the best performance for the CircularStatE Module, achieving a HOTA score of 24.107%.

2.3. More Analyses of Hyperparameters

Analysis of Impacts of Training Epochs. We further analyzed the variations observed across different epochs by selecting the same parameters (*i.e.*, track initialization threshold of 0.55 and track update threshold of 0.45). The experiments were conducted on the validation dataset of JRDB [8], with model weights saved every 5 epochs, and inference was performed at the end. The results are presented in Tab. 4. As shown in the table, different epochs have a noticeable impact on the final HOTA metric. When the epoch was set to 100, the best HOTA value of 25.12% was achieved, with results from other epochs slightly lower than this value. Overall, the results demonstrate that OmniTrack exhibits strong robustness and consistent performance across different epochs.

Analysis of FlexiTrack Instance Noise. FlexiTrack Instance (Sec. 3.3 in the main text) plays a crucial role in assisting the detection module to quickly locate targets in panoramic field-of-view scenarios and establish temporal associations between them. A key aspect of its performance is the initialization phase, where the selection of motion noise can significantly influence the overall tracking results. To investigate this, we analyze the impact of different motion noise levels on FlexiTrack Instance's performance on the validation set of JRDB [8], as presented in Tab. 5. From the table, it is evident that varying motion noise levels have

Exp.	Epoch	HOTA↑	IDF1↑	OSPA↓	MOTA↑
①	80	24.16	25.84	**0.93**	31.04
②	85	25.05	27.29	**0.93**	33.74
③	90	24.70	26.85	**0.93**	33.09
④	95	24.95	27.31	**0.93**	31.32
⑤	105	24.99	27.25	**0.93**	33.05
⑥	110	25.00	27.20	**0.93**	32.83
⑦	115	24.70	27.11	**0.93**	31.75
⑧	100	**25.12**	**27.42**	**0.93**	**34.99**

Table 4. Analysis of the impact of epochs on performance. Analysis of the performance impact of the OmniTrack$_{E2E}$ method across different epochs, with other parameters held constant.

Exp.	Noise	HOTA↑	IDF1↑	OSPA↓	MOTA↑
①	0.1	19.72	20.65	0.95	28.63
③	0.8	24.32	26.28	**0.93**	34.88
③	1.0	23.61	25.84	**0.93**	33.12
④	0.5	**25.12**	**27.42**	**0.93**	**34.99**

Table 5. Ablation of FlexiTrack Instance noise. The noise mentioned here refers to the one applied to the anchor (in Equation 6 of the main text), while the feature vector remains unchanged.

a notable effect on the final HOTA score. Specifically, a motion noise value of 0.5 improves performance, leading to a significant boost in tracking accuracy.

2.4. More Analyses of MOT Datasets

To visually assess the overall performance of existing state-of-the-art methods on panoramic MOT datasets, we compare the pinhole-based MOT17 [10] and DanceTrack [12] datasets with the panoramic datasets JRDB [8] and QuadTrack. As shown in Figure 1, MOTRv2 [18] achieves a HOTA of 73.4% on DanceTrack [12] but only 18.22% on JRDB [8], representing a decrease of 55.18%. Similarly, ByteTrack [17] achieves 63.1% HOTA on MOT17 [10] but only 20.66% on QuadTrack, a drop of 42.44%. Overall, the HOTA on panoramic datasets is approximately 30% lower than on pinhole-based datasets. More importantly, OmniTrack significantly outperforms existing SOTA methods on both panoramic datasets, marking a substantial advancement in the field of panoramic multi-object tracking.

3. Reproduction of state-of-the-art Methods.

Due to the absence of existing performance records for SOTA methods on the JRDB and QuadTrack datasets, all comparative experiments in this paper were independently reproduced. In the reproduction process, we prioritized using the official source code, provided it was executable. The parameter selection was based on the recommendations in the original papers, aiming to achieve optimal performance on both the JRDB and QuadTrack datasets.

3.1. Methods for the E2E Paradigm.

TrackFormer. To reproduce the TrackFormer method [9], we utilized the official source code (link) and applied it to both JRDB [8] and QuadTrack datasets. Our implementation uses COCO pre-trained weights from Deformable DETR [19], incorporating iterative bounding box refinement to enhance tracking accuracy. The model is trained on a single GPU with a batch size of 2. To adapt the model for JRDB [8] and QuadTrack datasets, we reformat the data to align with the MOT20 format [5], which is a widely used format in multi-object tracking challenges. Training is conducted for 30 epochs, with an initial learning rate of 2×10^{-4}. The learning rate is decayed by a factor of 10 every 10 epochs, as per the official guidelines. All other parameters remain unchanged, using the default values.

MOTR. In reproducing the MOTR method [16], we encountered challenges when training with the weights originally used in the TrackFormer method [9]. As a result, we opted to train the model on the JRDB dataset [8] using pre-trained weights from the MOT17 dataset [10], which is specifically designed for multi-object tracking tasks. The model is fine-tuned on a single GPU with a batch size of 1. To adapt the model to the JRDB dataset [8], we modified the data format to match the DanceTrack format [12]. This format adaptation ensures compatibility with the input requirements of the MOTR framework [16]. The model is trained for 25 epochs to ensure model convergence, with an initial learning rate of 2×10^{-4}. Based on the official source code (link) and our experience, the learning rate is reduced by a factor of 10 every 5 epochs. All other parameters were retained at their default values, as per the official guidelines.

MOTRv2. The pre-trained weights are identical to those used in TrackFormer [9]. The model is trained on a single GPU with a batch size of 1. To adapt the model for JRDB [8] and QuadTrack datasets, we convert the data to the DanceTrack format [12]. Since MOTRv2 [18] is highly dependent on detection results, we use ground truth detections for the training set to ensure optimal tracking performance. For the test set, to maintain fairness, we generate detection results using our own detector. The training procedure spans 15 epochs for JRDB [8] and 25 epochs for QuadTrack, after which the model ceases to converge. The initial learning rate is set to 2×10^{-4} with a decay factor of 10 every 5 epochs, in alignment with the settings used in MOTR [16]. All other parameters were retained at their default values, as specified in the official source code (link).

3.2. Methods for the TDB Paradigm.

HybridSORT. In reproducing the HybridSORT method [15] on both JRDB [8] and QuadTrack datasets, we utilized the official source code (link). HybridSORT offers two variants: an appearance-based version and an appearance-free version. For all experiments presented in this paper, the appearance-free version of HybridSORT was employed. For parameter selection, consistent values were applied across both JRDB [8] and QuadTrack datasets: `track_thresh` was set to 0.6 and `iou_thresh` was set to 0.15, in alignment with the settings used in the DanceTrack dataset [12]. All other parameters were kept at their default values, as specified in the official implementation.

SORT. As a pioneering approach in the TBD paradigm, the SORT method [2] has multiple implementation versions. However, due to the age of the original source code, it has been deprecated. In this paper, we chose to reproduce the SORT method based on the HybridSORT [15] source code (link). For both JRDB [8] and QuadTrack datasets, we set `track_thresh` to 0.6 and `iou_thresh` to 0.3, in alignment with the settings used for the SORT method on the DanceTrack dataset [12]. All other parameters were retained at their default values, as per the official guidelines.

DeepSORT. In the comparative experiments of this paper, we encountered compatibility issues with the DeepSORT [14] source code repository, which was not compatible with Torch models, complicating the reproduction process. As a result, we chose to reproduce the DeepSORT algorithm using the code from HybridSORT [15]. It is important to note that DeepSORT is an appearance-based tracking method, which, in theory, requires the separate training of the appearance module for both JRDB [8] and QuadTrack datasets. However, due to the lack of explicit guidance on training the appearance weights, we used the pre-trained appearance weights provided in the source code, specifically the `googlenet_part8_all_xavier_ckpt_56.h5` checkpoint. All other parameters were retained at their default values and were not modified.

ByteTrack & OC-SORT. In reproducing ByteTrack [17] and OC-SORT [3], we chose to use their official source code to ensure consistency and accuracy. All parameter settings were directly taken from the official demo configurations, which were specifically designed to optimize performance. These settings were applied uniformly across both JRDB and QuadTrack datasets to maintain a fair comparison. This approach allows for a reliable evaluation of the performance of both tracking algorithms on our datasets while adhering to the original implementation guidelines.

BoT-SORT. BoT-SORT [1] is a tracker in the TBD paradigm that integrates multiple techniques, including the use of appearance features. For both JRDB [8] and QuadTrack datasets, we trained the appearance feature model using Fast-ReID [7]. All other parameters were retained as specified in the original BoT-SORT source code (link), ensuring consistency with the default configuration.

3.3. YOLO11 Detection

In the TBD paradigm of tracking, the performance heavily depends on the detector's results. We selected the best detector in the YOLO series [11], YOLO11 [13], as the baseline for comparison. To enhance the perception capability, we selected the YOLO11 series model with the largest number of parameters, the YOLO11-X [13], for training. The training configuration consisted of 100 epochs, an image size of 960, and a batch size of 8, with all other settings maintained at their default values. Upon completion of the training, the model weights from the best-performing checkpoint, `best.pt`, were used to infer the images in the test set. Detection results with confidence scores greater than the threshold 0.1 were retained and subsequently provided as input to the tracker in the TBD paradigm.

4. Discussion

4.1. Societal Impacts

The OmniTrack framework is promising to enhance the safety and reliability of autonomous systems by improving Multi-Object Tracking (MOT) in panoramic settings, which is essential for applications such as self-driving cars and robots. Its ability to process panoramic fields of view while mitigating distortions ensures robust performance in dynamic, real-world environments. These advancements have the potential to benefit a wide range of industries, particularly in navigation for individuals with visual impairments, drone-assisted rescue, and hazardous object detection. Furthermore, the development of the QuadTrack dataset, designed for high-speed sensor motion and panoramic field-of-view applications, fills a critical gap in available resources. Aim to make both the dataset and the associated code publicly available, we intend to accelerate progress in the field of omnidirectional multi-object tracking, ultimately advancing the safety, efficiency, and inclusivity of automated systems in everyday life. Yet, it is inevitable that the deep model exhibits some false positives and negatives, and its practical deployment must account for the inherent uncertainty of deep neural networks. Additionally, while the technology is intended for benign applications, there exists a small risk of misuse, including potential military applications, and it may not be suitable for privacy-sensitive environments.

4.2. Limitations and Future Work

Although OmniTrack shows strong potential in the field of panoramic image tracking, it still has some limitations. While it does not exhibit ID confusion when targets are severely occluded, track loss can still occur in such scenarios. Future work could focus on addressing target occlusion, with one promising solution being multi-sensor fusion, such as integrating point cloud depth information to mitigate occlusion. This approach could extend 2D tracking to 3D tracking. Additionally, employing multiple agents that collaborate and share sensor information may enhance tracking performance, ultimately reducing track loss caused by occlusion and improving overall system robustness.

5. Visualization

MOT20. OmniTrack is a MOT framework specifically designed for panoramic FoV, facilitating target localization and association across distorted and panoramic FoV images. Unlike pinhole cameras, where objects tend to be denser, panoramic images typically feature more sparsely distributed targets. To intuitively demonstrate OmniTrack's performance in dense pedestrian scenarios, we visualize its tracking results on sequence 07 of the MOT20 test set [5], as shown in Fig. 3. The results indicate that OmniTrack successfully tracks most targets; however, it struggles with particularly small or heavily occluded objects, such as the one next to ID 20. The primary challenge stems from the limited training data in MOT20 [5], which contains only 4 sequences, posing a significant challenge for $OmniTrack_{Det}$. In future work, we aim to enhance tracking performance in dense target scenarios.

QuadTrack & JRDB. We visualize the final tracking results on the JRDB [8] and QuadTrack datasets, as shown in Fig. 5 and Fig. 6. In these images, red arrows highlight instances where trajectories were lost and not correctly tracked, while yellow arrows indicate identity confusion, leading to ID switches. In Fig. 5, for the JRDB dataset [8], we observe that OmniTrack accurately tracks objects, even in scenes with a large number of people, without any ID switches or trajectory losses. In contrast, ByteTrack [17] and SORT [2] both exhibit trajectory losses, while OC-SORT [3] experiences multiple ID switches. In Figure 6, for the QuadTrack dataset, the tracking of cyclists in the foreground remains intact, while OC-SORT, ByteTrack, and SORT all suffer from trajectory loss at frame 247. These examples demonstrate OmniTrack's superior recall ability, further validating the effectiveness of our feedback mechanism and the FlexiTrack Instance in accurately maintaining targets in panoramic-FoV scenarios.

References

[1] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. BoT-SORT: Robust associations multi-pedestrian tracking. *arXiv preprint arXiv:2206.14651*, 2022. 4

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *ICIP*, 2016. 4, 5, 8, 9

[3] Jinkun Cao, Jiangmiao Pang, Xinshuo Weng, Rawal Khirodkar, and Kris Kitani. Observation-centric SORT: Rethinking SORT for robust multi-object tracking. In *CVPR*, 2023. 4, 5, 8, 9

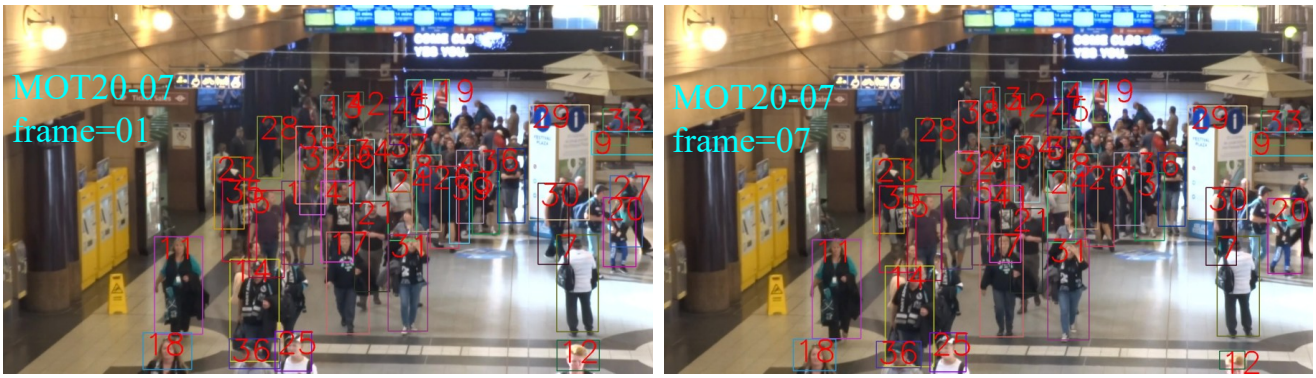

Figure 3. Visualizing the tracking performance of the OmniTrack method in dense crowds (MOT20 [5]).

[4] CVAT.ai. Computer vision annotation tool (CVAT). https://github.com/cvat-ai/cvat, 2024. Accessed: 2024-11-10. 1

[5] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian D. Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. MOT20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*, 2020. 4, 5, 6

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

[7] Lingxiao He, Xingyu Liao, Wu Liu, Xinchen Liu, Peng Cheng, and Tao Mei. FastReID: A pytorch toolbox for general instance re-identification. *arXiv preprint arXiv:2006.02631*, 2020. 4

[8] Roberto Martín-Martín, Mihir Patel, Hamid Rezatofighi, Abhijeet Shenoi, JunYoung Gwak, Eric Frankel, Amir Sadeghian, and Silvio Savarese. JRDB: A dataset and benchmark of egocentric robot visual perception of humans in built environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2023. 3, 4, 5, 8

[9] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. TrackFormer: Multi-object tracking with transformers. In *CVPR*, 2022. 4

[10] Anton Milan, Laura Leal-Taixé, Ian D. Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016. 1, 3, 4

[11] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 5

[12] Peize Sun, Jinkun Cao, Yi Jiang, Zehuan Yuan, Song Bai, Kris Kitani, and Ping Luo. DanceTrack: Multi-object tracking in uniform appearance and diverse motion. In *CVPR*, 2022. 3, 4

[13] Ultralytics. YOLO vision. https://github.com/ultralytics/ultralytics, 2024. Accessed: 2024-11-10. 5

[14] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *ICIP*, 2017. 4

[15] Mingzhan Yang, Guangxin Han, Bin Yan, Wenhua Zhang, Jinqing Qi, Huchuan Lu, and Dong Wang. Hybrid-SORT: Weak cues matter for online multi-object tracking. In *AAAI*, 2024. 4

[16] Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. MOTR: End-to-end multiple-object tracking with transformer. In *ECCV*, 2022. 4

[17] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. ByteTrack: Multi-object tracking by associating every detection box. In *ECCV*, 2022. 3, 4, 5, 8, 9

[18] Yuang Zhang, Tiancai Wang, and Xiangyu Zhang. MOTRv2: Bootstrapping end-to-end multi-object tracking by pre-trained object detectors. In *CVPR*, 2023. 3, 4

[19] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *ICLR*, 2021. 4

Figure 4. Examples of the established QuadTrack dataset. The QuadTrack dataset features a variety of scenes, including different campuses, streets, and low-light environments, with machine-generated labels for each scenario. These labeled scenes demonstrate the diversity and complexity of the dataset, offering insights into the challenges of multi-object tracking across different real-world contexts.

OmniTrack (ours)

ByteTrack

OC-SORT

SORT

Figure 5. Visualization on the public JRDB dataset [8]. The visualization compares the performance of OmniTrack, SOTA [2], ByteTrack [17], and OC-SORT [3] methods on the JRDB validation set. The red arrows in the figures indicate instances where the trajectories were not correctly tracked, leading to tracking losses, while yellow arrows highlight cases of track ID confusion, indicating ID switches.

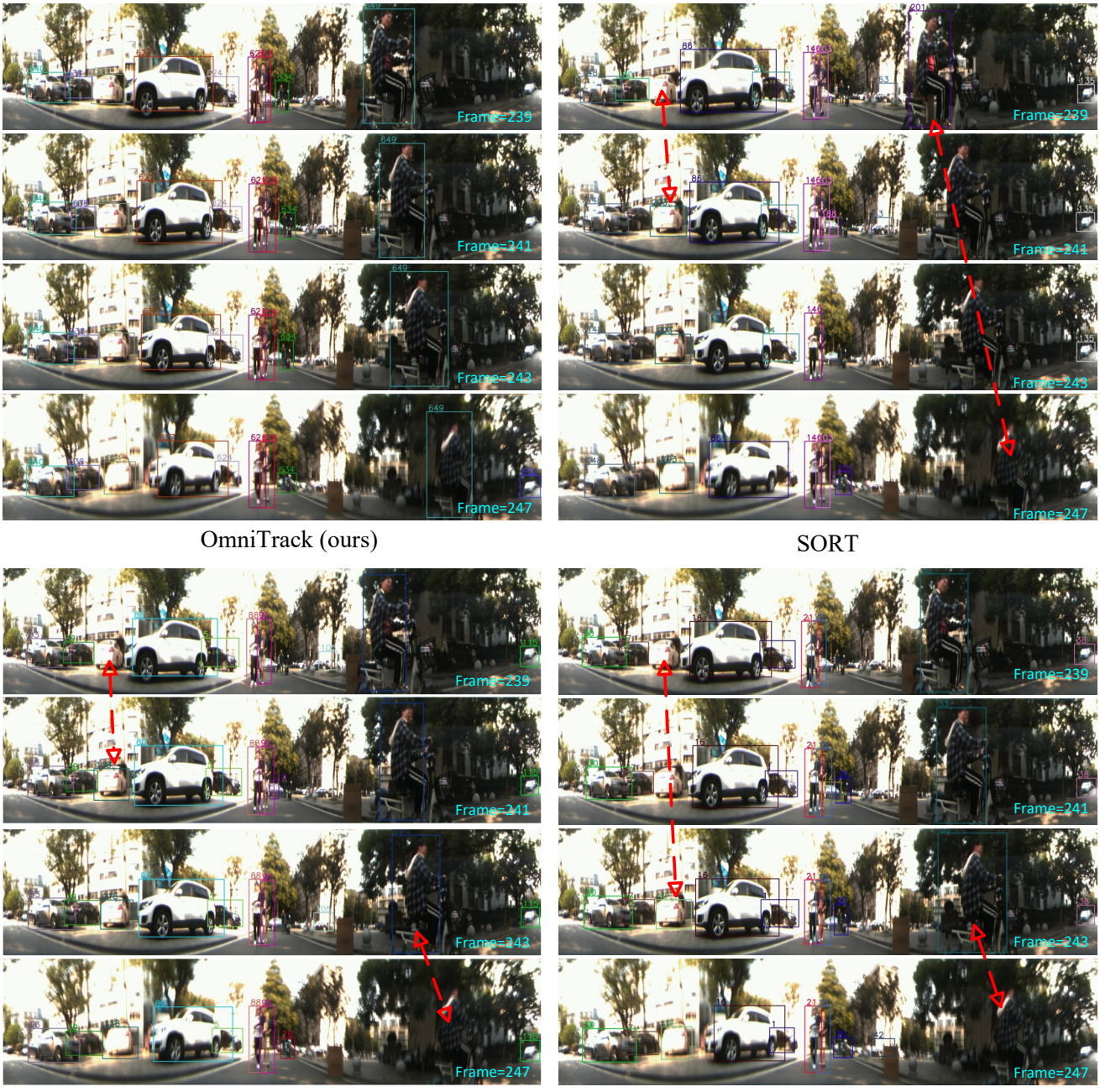

Figure 6. Visualization comparison on the established QuadTrack dataset. The visualization compares the performance of OmniTrack, SOTA [2], ByteTrack [17], and OC-SORT [3] methods on the QuadTrack test set. The red arrows in the figures indicate instances where the trajectories were not correctly tracked, leading to tracking losses.