From Detection to Association: Learning Discriminative Object Embeddings for Multi-Object Tracking

Supplementary Material

A. Overview

In the supplementary material, we primarily:

1. Present design rationale and implementation details in Sec. B.
2. Present experimental details including dataset processing, training strategies, and loss function in Sec. C.
3. Provide additional experimental analysis and visualization results in Sec. D.

B. FDTA Details

In this section, we present the design rationale and implementation details for the Spatial Adapter (Sec. B.1), Temporal Adapter (Sec. B.2), and Identity Adapter (Sec. B.3).

B.1. Spatial Adapter Design

B.1.1. Depth Extractor

The depth extractor fuses pyramid features from the backbone $\boldsymbol{F}_V$. Similar to the visual branch, we use three feature levels with spatial strides of 8, 16, and 32 (denoted as $\mathbf{f}_8$, $\mathbf{f}_{16}$, and $\mathbf{f}_{32}$) to capture both fine-grained spatial details and global scene context for depth estimation.

To derive the dense features $\boldsymbol{F}_{dense}$, we first project features from each scale to a unified dimension $c = 256$ using 1×1 convolutional layers. The coarser-scale features at strides of 16 and 32 are then upsampled to the finest resolution via bilinear interpolation and averaged:

$$\boldsymbol{F}_{\text{avg}} = \frac{1}{3}(\mathbf{f}_8 + \mathbf{f}_{16}^{\uparrow} + \mathbf{f}_{32}^{\uparrow}) \tag{10}$$

The averaged features $\boldsymbol{F}_{\text{avg}}$ are then processed through two 3×3 convolutional blocks to produce the final dense features $\boldsymbol{F}_{dense}$ for depth prediction.

B.1.2. Depth Discretization

Inspired by MonoDETR [53], we formulate depth prediction as a classification task for stable training. This requires discretizing the continuous pseudo depth values from Video Depth Anything [8] into depth bins, each representing a depth range. We employ Linear-Increasing Discretization (LID) to partition these bins.

We compute the bin size as:

$$\text{bin\_size} = \frac{2(d_{\max} - d_{\min})}{K(1+K)} \tag{11}$$

For the K foreground bins, the depth values are computed as:

$$b_i = (i + 0.5)^2 \cdot \frac{\text{bin\_size}}{2} - \frac{\text{bin\_size}}{8} + d_{\min} \tag{12}$$

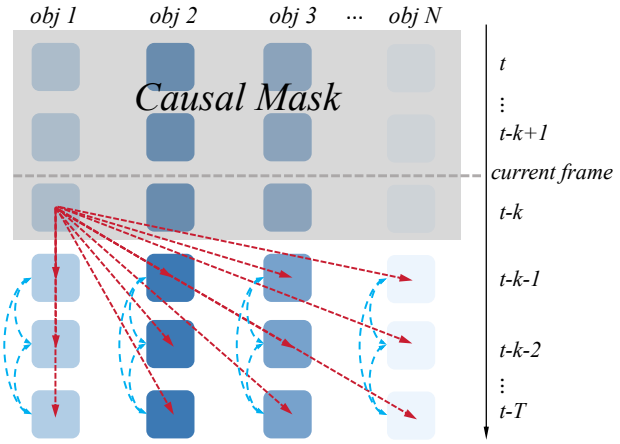

Figure 8. Object embedding interaction in ID prediction and TA. Red arrows represent the interaction in ID prediction, where objects at the current frame query historical embeddings for identity matching. Blue arrows represent the interaction in TA, where embeddings within the same trajectory interact across frames to enrich temporal context.

where $i = 0, 1, \ldots, K-1$. For the last bin corresponding to background, we set $b_K = d_{\max}$. Here $d_{\min} = 10^{-3}$ and $d_{\max} = 256$ define the depth range. Unlike uniform binning, LID allocates more bins to near depth ranges where tracking is more critical.

B.1.3. Depth Prediction Head

For depth prediction, the depth head predicts a probability distribution over the $K+1$ bins for each pixel, which is then converted to continuous depth values. Specifically, a single-layer 1×1 convolutional head maps the dense features $\boldsymbol{F}_{dense}$ to $(K+1)$ channels. After softmax, we obtain the per-pixel probability distribution $\boldsymbol{d}$. The final continuous depth map $\hat{\boldsymbol{d}}$ is computed via weighted summation:

$$\hat{\boldsymbol{d}} = \sum_{i=0}^{K} d_i \cdot b_i \tag{13}$$

where d_i is the predicted probability for the i-th bin and b_i represents the depth value of that bin.

B.2. Temporal Adapter Design Motivation

As illustrated in Figure 8, the ID prediction assigns identities via cross-attention where objects at current frame $t-k$ query all historical object embeddings from preceding frames $[t-k-1, \ldots, t-T]$ to retrieve the closest match (red

DanceTrack BFT SportsMOT

Figure 9. Visual examples from different datasets used in our experiments.

arrows). However, each historical embedding is independently encoded without inter-frame interaction, containing only information from its own frame. To address this limitation, we propose the Temporal Adapter (TA) to enable interaction among historical embeddings before ID prediction. Through temporal modeling across historical frames (blue arrows), each object embedding aggregates information from its trajectory history, thereby enriching embeddings with temporal context for more discriminative identity matching.

B.3. Identity Adapter Design

MoCo [19] demonstrates that more positive pairs improve contrastive learning. However, previous methods [12, 36] typically perform contrastive learning only between consecutive frames, yielding limited positive pairs per embedding. We expand positive pair selection by leveraging all embeddings across the entire training batch from multiple trajectories and frames. Given an identity appearing in M frames, any two from different frames form a positive pair, yielding $M(M-1)/2$ pairs. This significantly increases training samples and provides richer supervision for learning discriminative features.

C. Experimental Details

This section provides detailed experimental configurations that complement the main text, including dataset processing (Sec. C.1), training implementation details (Sec. C.2), and loss function details (Sec. C.3).

C.1. Dataset Processing

We evaluate our method on three diverse benchmarks: DanceTrack, SportsMOT, and BFT, strictly following standard protocols [14–16, 32, 52, 56] with official train/val/test splits. Specifically, DanceTrack consists of 40 training sequences, 25 validation sequences, and 35 test sequences; SportsMOT contains 45 training sequences, 45 validation sequences, and 150 test sequences; BFT provides 45 training sequences, 25 validation sequences, and 36 test sequences. Visual examples from these datasets are provided in Fig. 9.

For training, following standard protocols [15, 16, 52, 56], we adopt a sequence-based sampling strategy where each training sample consists of a trajectory with length $T = 30$. To enable robust learning of motion dynamics across varying speeds, we sample frames with random temporal intervals between 1 and 4, rather than using fixed consecutive frames. The same trajectory length T is maintained during inference to ensure consistency.

For input data, we use original images with a resolution of 1920×1080, and generate corresponding depth maps at the same resolution offline using Video Depth Anything [8] with the ViT-L backbone. During training, consistent data augmentation is applied to both RGB images and depth maps to guarantee strict spatial alignment.

C.2. Training Details

We adopt several key strategies for efficient and stable training. Specifically, we inject sinusoidal position encodings [40] into the Temporal Adapter to capture temporal order. Additionally, to mitigate the instability of DETR-based detectors in early training stages, we adopt a warm-up strategy for the Identity Adapter. We train for 11 epochs in total, and disable the contrastive loss in the first epoch to ensure learning from reliable object features.

C.3. DETR Loss Function

Following standard practices in Deformable DETR [61], the detection loss combines three components:

$$\mathcal{L}_{\text{det}} = \lambda_{\text{cls}}\mathcal{L}_{\text{cls}} + \lambda_{\text{bbox}}\mathcal{L}_{\text{bbox}} + \lambda_{\text{giou}}\mathcal{L}_{\text{giou}}, \tag{14}$$

Bounding box regression consists of an L_1 loss measuring coordinate differences and a GIoU loss capturing spatial overlap:

$$\mathcal{L}_{\text{bbox}} = \|b_{\text{pred}} - b_{\text{gt}}\|_1, \quad \mathcal{L}_{\text{giou}} = 1 - \text{GIoU}(b_{\text{pred}}, b_{\text{gt}}). \tag{15}$$

Classification is optimized using focal loss:

$$\mathcal{L}_{\text{cls}} = -\alpha_t(1-p_t)^{\gamma}\log(p_t), \tag{16}$$

where p_t is the estimated probability for the target class, α_t weights different classes, and γ controls the focus on

Figure 10. Pseudo depth labels visualization on DanceTrack, SportsMOT, and BFT.

hard examples. We keep the same hyperparameters as Deformable DETR [61]: $\lambda_{\text{cls}} = 2.0$, $\lambda_{\text{bbox}} = 5.0$, $\lambda_{\text{giou}} = 2.0$, $\alpha_t = 0.25$, and $\gamma = 2$.

D. Additional Experimental Analysis

D.1. Pseudo Depth Label Generation Analysis

D.1.1. Computational Cost and Storage Requirements

We generate pseudo depth labels offline using Video Depth Anything [8], eliminating online inference overhead. For the entire training set, the inference takes approximately 1 hour on an H200 GPU for DanceTrack, 0.6 hours for SportsMOT, and 0.4 hours for BFT. The storage requirements are reasonable, with approximately 8.4 GB for DanceTrack, 3.4 GB for SportsMOT, and 1.4 GB for BFT. This one-time cost is negligible compared to the training time, and the generated depth labels can be reused across multiple experiments.

D.1.2. Pseudo Depth Quality

As shown in Fig. 10, we visualize the pseudo depth labels generated by Video Depth Anything [8] across three diverse datasets. The generated depth maps exhibit high quality, maintaining consistent depth estimation both within individual frames and across temporal sequences. Additionally, they effectively distinguish objects at varying spatial positions. This provides reliable supervision for tracking and discriminative cues for embedding enhancement.

D.2. Discussion on Performances of Tracking-by-Detection Methods

In comparison with SOTA methods (see Tabs. 1 to 3), tracking-by-detection methods [1, 30, 35] achieve high MOTA and DetA scores compared to end-to-end methods [15, 16, 52, 56], but show relatively lower performance in metrics such as AssA, IDF1, and HOTA. This is because different metrics emphasize different aspects of tracking capability.

Understanding what each metric measures is key to proper evaluation. MOTA heavily weighs detection errors, while DetA directly evaluates detection accuracy. Tracking-by-detection methods naturally excel in these detection-focused metrics by leveraging powerful standalone detectors like YOLOX [17]. However, the core challenge in tracking is maintaining consistent identities across frames, not just detection. Metrics such as HOTA [29], which equally weights detection and association accuracy, and association-focused metrics (AssA, IDF1) that specifically evaluate the ability to maintain consistent identities, better reflect tracking capability. Our method achieves significant improvements on these association metrics, demonstrating strong tracking performance.

D.3. Additional Ablations on Spatial Adapter

D.3.1. Depth Encoding Layer Design

To integrate depth information into object embeddings, we investigate the optimal placement of the depth cross-attention layer within the decoder. As shown in Tab. 9, we insert the depth cross-attention layer (*Depth*) at different positions in each decoder block of the standard DETR decoder, which contains self-attention (*Self*) and visual cross-attention (*Vision*). We compare three placements: before self-attention (Row 2), between self-attention and visual cross-attention (Row 3), and after visual cross-attention (Row 4), with Row 1 showing the performance without

Table 9. Ablation study on depth encoding layer designs in SA. *Self*, *Vision*, and *Depth* denote self-attention, visual cross-attention, and depth cross-attention layers.

Architecture	HOTA↑	IDF1↑	AssA↑	MOTA↑	DetA↑
Self → *Vision*	69.4	74.5	60.2	90.6	80.0
Depth → *Self* → *Vision*	68.2	72.4	58.1	90.4	80.2
Self → *Depth* → *Vision*	69.3	73.7	59.6	**91.2**	80.7
Self → *Vision* → *Depth*	**70.2**	**74.8**	**61.2**	90.9	**80.7**

Table 10. Ablation study on foreground weighting in SA.

Setting	HOTA↑	IDF1↑	AssA↑	MOTA↑	DetA↑
w/o Foreground Weighting	70.7	75.7	61.6	**91.4**	**81.3**
w/ Foreground Weighting	**71.7**	**77.2**	**63.5**	91.3	81.0

depth. Results show that placing depth cross-attention after visual cross-attention (*Self* → *Vision* → *Depth*) achieves the best performance with 0.8% HOTA improvement. Therefore, we adopt this configuration in our Spatial Adapter.

D.3.2. Foreground Weighting Factor

As shown in Tab. 10, we evaluate foreground weighting in depth loss, which assigns larger weights to pixels within object bounding boxes during training. This strategy encourages the model to focus on learning accurate depth for foreground objects, which is more critical for tracking. The results show 1.0% HOTA and 1.5% IDF1 improvement, demonstrating its effectiveness.

D.4. Temporal Adapter Design Ablation

To verify that TA effectively leverages temporal information, we evaluate different trajectory history lengths as shown in Tab. 11, where the blue superscripts denote the performance gains from using TA. We observe that TA brings larger improvements with longer history. Specifically, at 5 frames, TA improves HOTA by +0.7%, while at 30 frames, the gain increases to +1.0% HOTA and +1.2% IDF1. This trend demonstrates TA's practical effectiveness. Longer sequences contain richer temporal information, and TA's attention mechanism enables each object embedding to aggregate information across all historical frames, enriching embeddings with comprehensive temporal context for better association. We adopt 30 frames as the default setting, balancing performance and training efficiency.

D.5. Additional Experimental Results

D.5.1. Depth Attention Visualization

To understand how SA leverages depth information, we visualize the attention maps of the depth cross-attention layer in the last decoder block, as shown in Fig. 11. For each target query (marked as white dots), the attention maps concentrate on foreground objects at similar depth values

Table 11. Ablation study on trajectory history length in TA. The blue superscripts denote the performance gains from using TA.

Length	w/o TA		w/ TA		Time (h/ep)
	HOTA↑	IDF1↑	HOTA↑	IDF1↑	
5	62.7	62.8	63.4$^{+0.7}$	62.8$^{+0.0}$	1.0
10	66.6	67.9	67.2$^{+0.6}$	69.2$^{+1.3}$	1.4
20	68.7	72.7	69.6$^{+0.9}$	73.7$^{+1.0}$	2.0
30	69.4	74.5	**70.4**$^{+1.0}$	**75.7**$^{+1.2}$	2.9

while suppressing background regions. By aggregating such depth-aware features, object queries obtain enriched embeddings with enhanced discriminativeness for better association.

D.5.2. Interaction Weight Visualization

We visualize the temporal interaction weight matrix in TA, as shown in Fig. 12. Each matrix shows how a tracked object queries its historical frames, where rows represent the query frame (current frame) and columns represent key frames (historical frames). The visualization result confirms that the proposed dual-mask strategy achieves its intended effect: The lower triangular structure results from the causal constraint, ensuring each frame only attends to past frames. Blue vertical stripes mark frames where the object is absent and correctly masked, showing that the mechanism handles undetected objects. Notably, the attention weights are not concentrated on adjacent frames but distributed across the full trajectory, indicating that TA effectively captures long-range temporal dependencies for comprehensive trajectory modeling.

D.5.3. More Tracking Results and Embedding Visualization

We provide additional tracking results and inter-object embedding similarity matrix visualizations on DanceTrack, SportsMOT, and BFT, as shown in Fig. 13. The visualizations reveal that baseline methods, MOTRv2 and MOTIP, exhibit high inter-object similarity in the embedding space, leading to tracking errors. In contrast, our method produces more discriminative embeddings by reducing inter-object similarity, achieving more stable tracking across diverse scenarios.

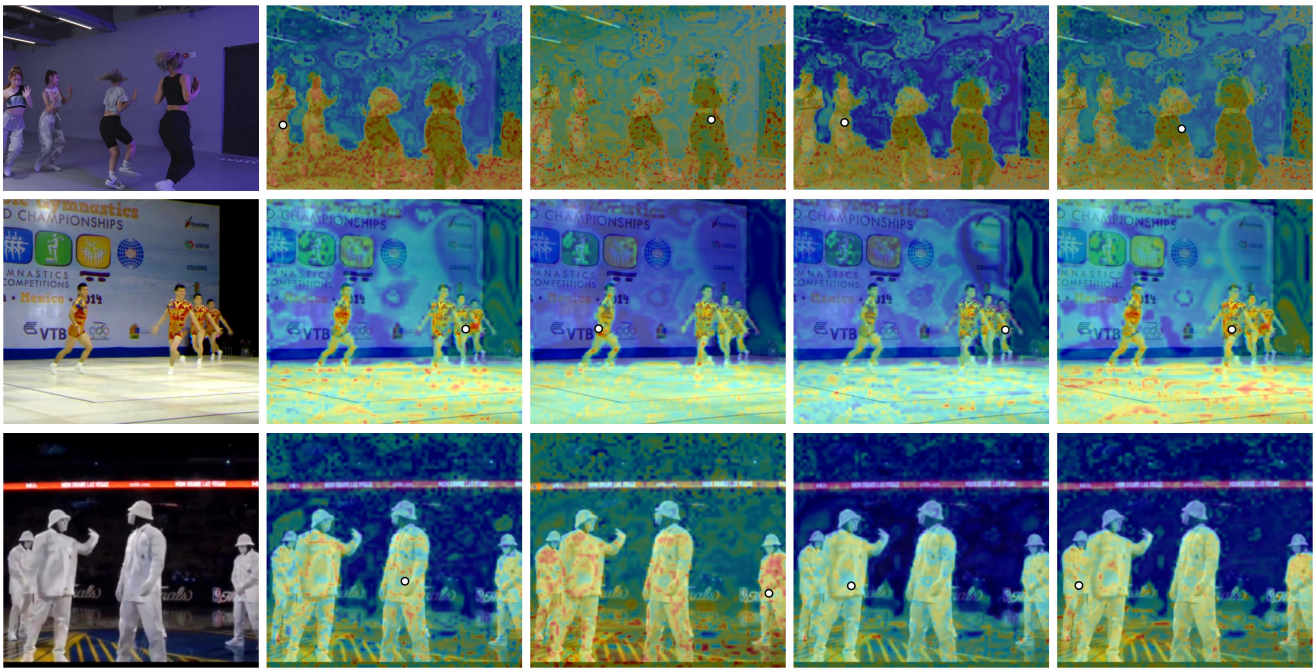

Figure 11. Visualizations of attention maps in SA. The first column denotes the input image, and the last four columns denote the attention maps of the target queries (denoted as white dots). Warmer colors indicate higher attention weights.

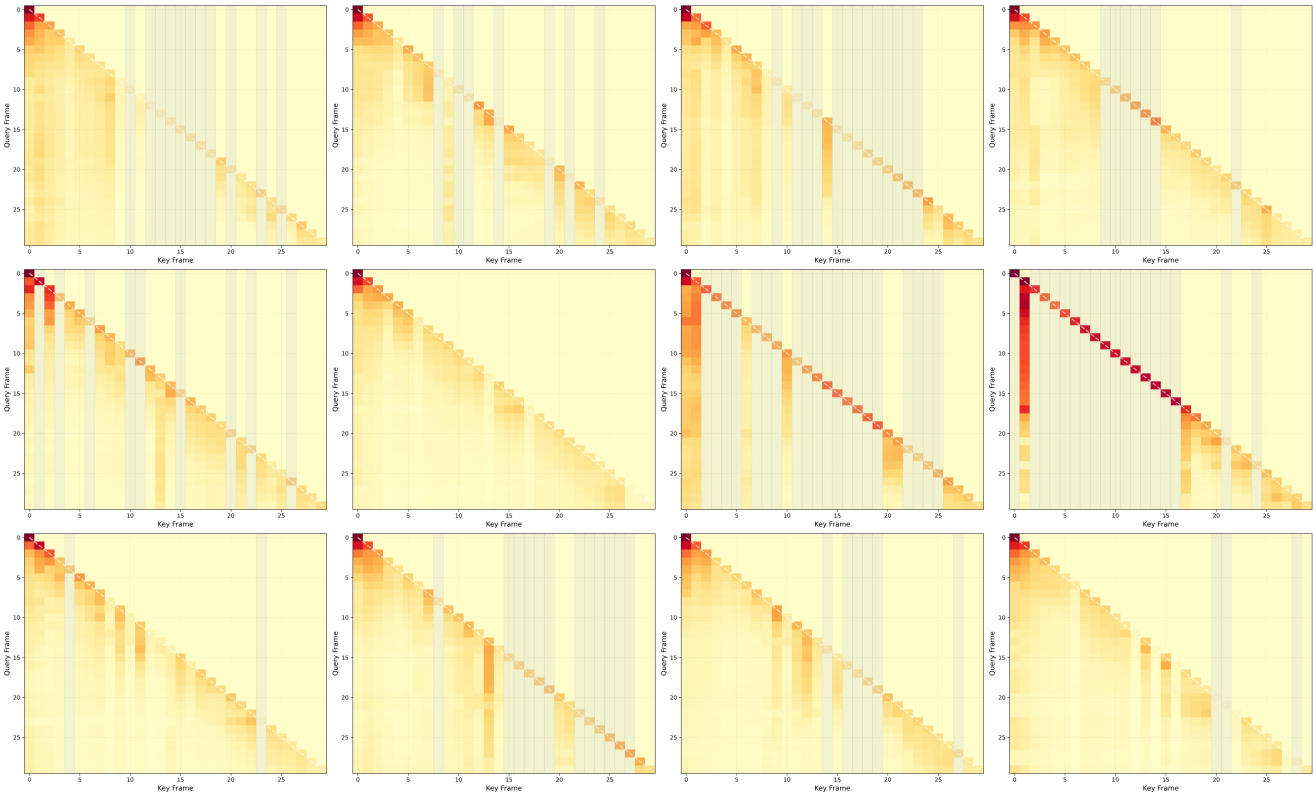

Figure 12. Visualizations of attention weight matrix in TA. Rows represent query frames (current frame) and columns represent key frames (historical frames). Warmer colors indicate higher attention weights. Blue vertical stripes indicate frames where the object is absent, and the lower triangular structure results from the causal mask.

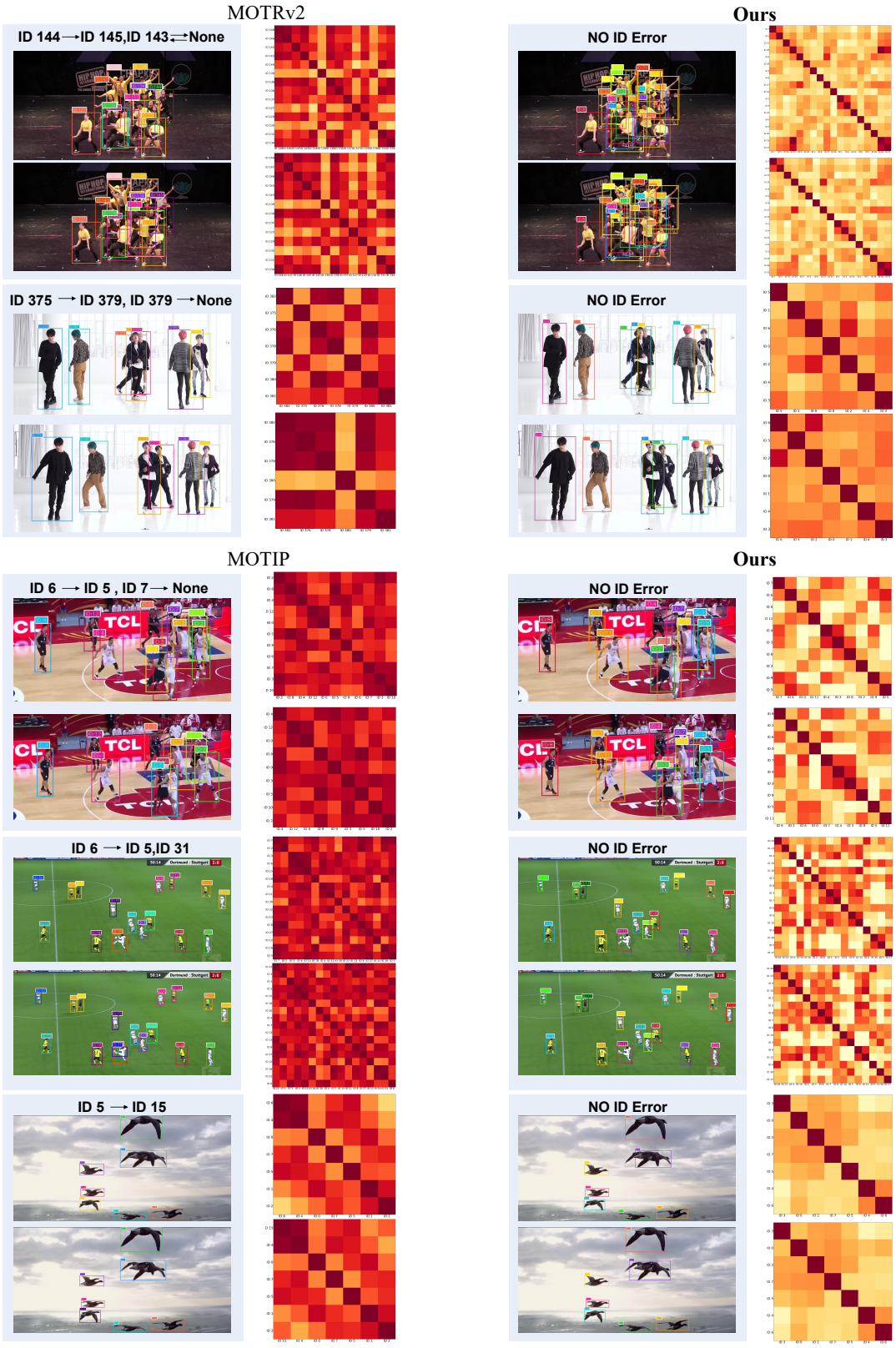

Figure 13. More tracking results visualization and inter-object embedding similarity matrix comparison on DanceTrack, SportsMOT, and BFT. Darker colors in the similarity matrix indicate higher similarity.