Hypergraph-State Collaborative Reasoning for Multi-Object Tracking

Supplementary Material

In this supplementary material, we describe more architecture details and training details of our HyperSSM approach in Section A. We validate the advantages of noise suppression and occlusion handling enabled by the hypergraph structure, and carry out a deep analysis in Section B.

A. Implementation details

A.1. Architecture details

Our architecture primarily consists of a Trajectory Embedding Module and a HyperSSM-based Motion Estimation Module, which can be viewed as a classic encoder-decoder framework. The complete motion estimation procedure is formally described in Algorithm 1. The trajectory embedding module is designed to embed the recent historical trajectories over a defined interval, providing contextual guidance for the decoder's motion estimation. The motion estimation module, integrating Hypergraph computation in to SSM structure, utilizes the motion state of previous frames, which are modeled as hypergraph correlations, to predict the motion result of the next frame. Inspired by TrackSSM [3], our framework decouples trajectory embedding module from the motion estimation module, this separation ensures a robust and adaptive handling of temporal dependencies in motion estimation.

Trajectory Embedding Module. We use the Mamba encoder [2] as our trajectory embedding module to embed the recent historical trajectories. Assuming we are processing the motion estimation of t-th frame, the historical trajectories over a defined interval $[t-L, t-1]$ across L frames, we encode this position information $P_i, i \in [t-L, t-1]$ into trajectory embeddings $\mathcal{T}_t$ through a multi-layer of HyperSSM block. These trajectory embeddings $\mathcal{T}_t$ are used as guidance for the motion estimation decoder.

Motion Estimation Module. The motion estimation module is designed to accurately regress the trajectory box from the previous frames to the current frame. The decoder is composed of 6 identical layers cascaded together, with each decoder layer containing a HyperSSM block and a FFN block. The trajectory embeddings $\mathcal{T}_t$ obtained from the trajectory embedding module are act as a hidden state and fed into the HyperSSM block to obtain the motion features. The motion features are passed through the FFN to estimate the motion result $\mathbf{P}_t$. Within the motion estimation, trajectory embeddings are applied to each layer. The output from the previous decoder layer serves as the input to next layer, enabling precise refinement of the trajectory over time. Additionally, the hidden state acts as a messenger, transmitting information between cascaded layers, allowing the trajectory to progressively regress toward the ground truth labels.

Tracks Association. Given the predicted track states P at frame t and the detection set $D[t]$, we first divide detections into three subsets: D_{high} (high-confidence and kept by NMS), D_{low} (low-confidence but retained by NMS), and D_{del} (high-confidence but suppressed by NMS). These subsets are combined to form a unified candidate pool. We then construct a cost matrix C between each predicted track P_i and each candidate detection, where the cost includes spatial distance (e.g., IoU-based or motion-based discrepancy) together with a small category-dependent penalty that reflects the relative reliability among $D_{\text{high}}, D_{\text{low}}, D_{\text{del}}$. Track–detection association proceeds iteratively: at each step we find the pair $(i, j) = \text{TrackBestPair}(C)$ that is simultaneously a row-minimum and column-minimum candidate, indicating that D_j is the most suitable detection for track P_i and vice versa. Once a pair is selected, we assign D_j to P_i and remove the corresponding row and column from C. This process repeats until the minimum remaining cost exceeds the association threshold τ_{assoc}.

Tracks Initialization. After association, detections that remain unassigned form the set U. As some of these detections may lie close to the existing tracks, we apply $\text{TAI\_NMS}(U, \text{LastPositions}(B))$ to suppress detections that heavily overlap with the latest positions of active tracks, ensuring that unnecessary or duplicated tracks are not created. The remaining detections in the filtered set U are then used to initialize new tracking hypotheses, where each detection $d \in U$ spawns a new track in B. This strategy reduces false positive track creation and keeps the tracker robust in cluttered scenes.

A.2. Training and Inference

Training Phase. During the training phase, we adopt the step-by-step linear training strategy base on [3]. We decompose the temporal process of bounding box regression into K linearly interpolated sub-steps to simplify complex motion transitions. Given a bounding box P_t at frame t and target P_{t+1}, we divide the regression into equal intervals $\Delta_t = \frac{(t+1)-t}{K}$ and constructs pseudo-labels $\{P_{t+k\Delta_t}\}_{k=1}^{K}$ through linear interpolation $\mathcal{I}(P_{t+1} \leftarrow P_t)$. Each of the motion estimation module layers then performs incremental refinement $P_{t+k\Delta_t} \rightarrow P_{t+(k+1)\Delta_t}$, enforcing collaborative learning across layers. This approach stabilizes nonlinear motion prediction by breaking it into equal linear sub-tasks, improves generalization for non-rigid deformations through gradual transitions, and enhances lost trajectory recall (evidenced by higher IDF1 scores) while maintaining the core intuition that flow features should guide all refinement steps

Algorithm 1: Motion estimation and association in our HyperSSM approach

Input: video frames F
Output: tracks B

Initialization:
    $L_f \leftarrow len(F)$; // Total length of the input video
    L ; // Window length
    $t = 0$; // Frame index
while $t < L_f$ **do**
    $D[t] = detector(F[t])$; // **detection** at frame t
    if $t == 0$ **then**
        $B[0] \leftarrow D[0]$; // Initialize tracks
        ;
    if $t < L$ **then**
        $P \leftarrow B[0 : t-1]$; // Use full history
    else
        $P \leftarrow B[t-L+1 : t-1]$; // Use windowed history
    $\mathcal{T} \leftarrow$ `TrajectoryEmbed`(P) ; // History **trajectory** embedding
    for $layer_i$ *in HyperSSM* **do**
        $P \leftarrow layer_i(P, \mathcal{T})$; // Cascaded motion estimation
    $D^h, D^l, D^d \leftarrow$ `SplitDetections`$(D[t])$; // Split by detection confidence
    $C \leftarrow$ `BuildCost`$\big(P,\ D^h \cup D^l \cup D^d\big)$; // Track-detection cost matrix
    while $\min(C) < \tau_{assoc}$ **do**
        $(i, j) \leftarrow$ `BestPair`(C) ; // **association:** best track-detection pair
        $B[t] \leftarrow B[t] \cup \{(P_i, D_j)\}$; // Update matched tracks
        $C \leftarrow$ `MaskRowCol`(C, i, j) ; // Invalidate used row/column
    $U \leftarrow$ `RemainingDets`(D^h, D^l, D^d) ; // Unmatched **detections**
    $U \leftarrow$ `SuppressOverlaps`$\big(U,$ LastBoxes$(B)\big)$; // Suppress overlaps with strong tracks
    for $d \in U$ **do**
        `InitTrack` (B, d) ; // **initialization** of new tracks

uniformly, differing from traditional end-to-end regression where layers independently predict P_{t+1}.

We use ground truth as well as pseudo labels obtained through linear interpolation to supervise HyperSSM. We take the smooth L1 loss and generalized intersection over union (GIoU) [4] between the estimation and ground-truth of all frames within the estimated sliding window:

$$\mathcal{L} = \lambda_1 \sum_{t}^{L} \mathcal{L}_{L1}(P_t, P_t^{gt}) + \lambda_2 \sum_{t}^{L} \mathcal{L}_{GIoU}(P_t, P_t^{gt}) \quad (1)$$

where L represents the length of a sliding window, λ_1 and λ_2 represent the weight coefficients for smooth L1 loss and GIoU loss, respectively, P_{gt} is the ground-truth labels.

Inference Phase. During inference, our tracker operates within the standard tracking-by-detection paradigm. Each incoming frame is first processed by the YOLOX [1] detector to obtain object candidates. For every active track, we gather its recent trajectory history and feed it into the HyperSSM module, which performs layer-wise motion estimation to refine the predicted states. These refined predictions are then used to associate tracks with the new detections through an iterative matching process that considers both spatial proximity and confidence cues. After association, any unmatched detections are examined to initialize new tracks, while unmatched tracks are either updated, kept alive for a short period, or removed depending on their temporal consistency. This unified inference pipeline enables HyperSSM to maintain stable and accurate trajectories even under complex motions or crowded conditions.

B. Additional experiments

Our HyperSSM uses motion-aware hyperedges to group objects with similar movement patterns, allowing it to analyze their collective behavior. This approach improves tracking in two ways: (1) it reduces errors in individual predictions by relying on more stable group motion patterns, where neighboring objects help correct anomalies through natural

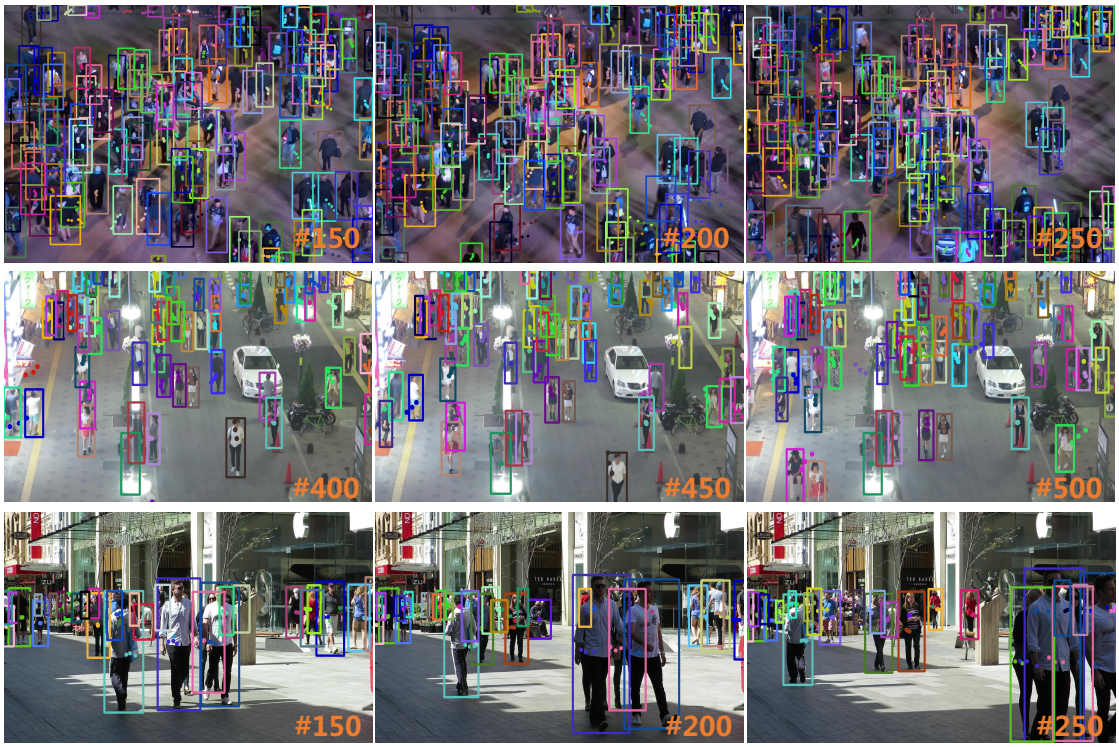

Figure 1. Qualitative tracking results in challenging scenarios. HyperSSM maintains stable associations in crowded scenes and preserves accurate motion reasoning, demonstrating strong robustness in complex MOT settings.

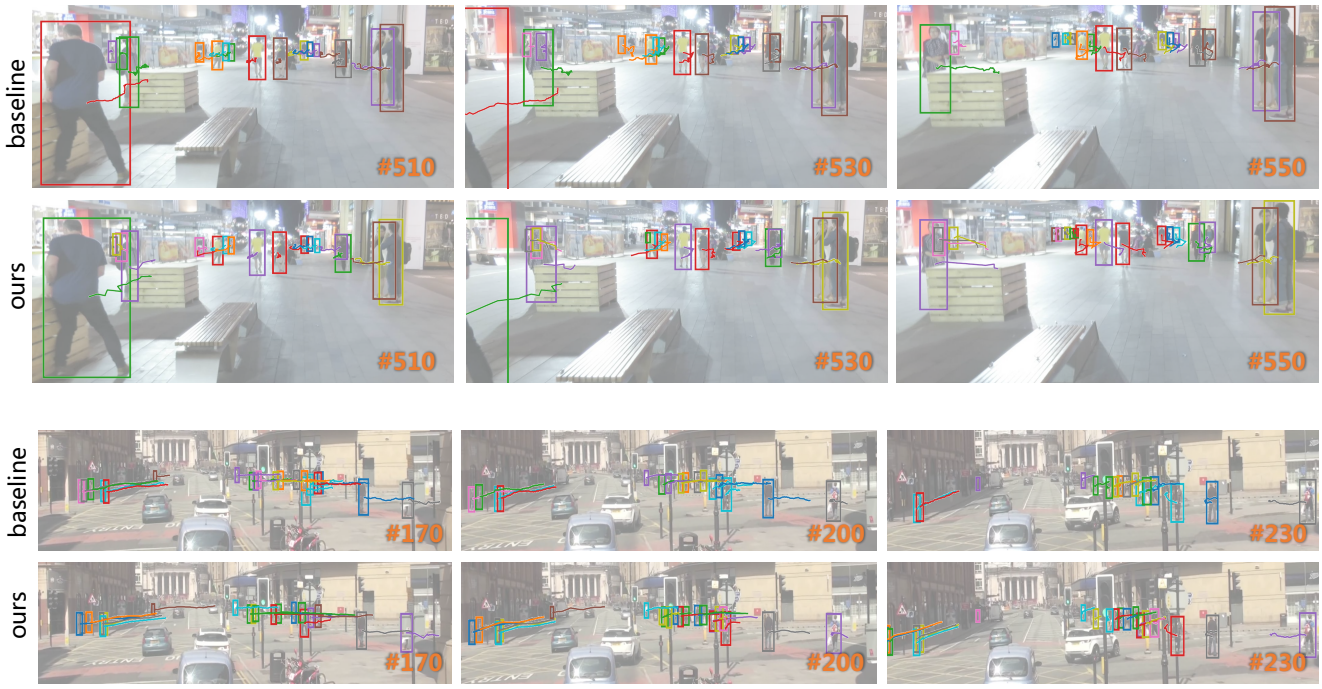

Figure 2. Visualization of occlusion handling comparison between baseline and our hypergraph-based method, showing robust trajectory recovery through collaborative motion reasoning under heavy occlusions.

physical constraints. (2) it maintains accurate tracking even during occlusions by estimating hidden objects' movements based on the observed motion of related objects in the same group. We conducted comprehensive experiments to validate both of these advantages.

B.1. Qualitative Results in Challenging MOT Scenarios

To further assess the robustness of our HyperSSM motion estimation module, we conduct evaluations on highly congested and visually complex scenes. As shown in Figure 1, the displayed frames contain dense crowds with frequent interactions, irregular trajectories, and substantial mutual occlusions. Despite these challenges, HyperSSM is able to maintain stable identity continuity by accurately capturing the underlying complex motion patterns of each target and by suppressing spurious associations induced by clutter. The visual results demonstrate that our model preserves clear target separation even when objects move unpredictably or overlap heavily, highlighting the strong adaptability and reliability of HyperSSM in real-world crowded scenarios.

B.2. Analysis on Handling of Occlusion

The Figure 2 demonstrates our occlusion handling performance where each subfigure compares baseline results (top) without hypergraph computation against our hypergraph-based approach (bottom), clearly showing that the baseline frequently loses targets and produces broken trajectories under heavy occlusion, highlighting the challenge of motion estimation in such scenarios, while our collaborative reasoning successfully maintains more complete trajectories and recovers occluded targets by leveraging inter-object relationships, proving our method's effectiveness in predicting invisible targets through object association and motion transfer between related objects.

References

[1] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. 2

[2] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. 1

[3] Bin Hu, Run Luo, Zelin Liu, Cheng Wang, and Wenyu Liu. Trackssm: A general motion predictor by state-space model. *arXiv preprint arXiv:2409.00487*, 2024. 1

[4] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union: A metric and a loss for bounding box regression. In *Proceedings of the CVPR*, pages 658–666, 2019. 2