SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

May 2026, Vol. 69, Iss. 5, 152105:1–152105:13
https://doi.org/10.1007/s11432-024-4849-2

TrackSSM: a general motion predictor using a state space model

Bin HU[1,2], Run LUO[3], Zelin LIU[2], Cheng WANG[2] & Wenyu LIU[2*]

[1]*Institute of Artificial Intelligence, Huazhong University of Science and Technology, Wuhan 430074, China*
[2]*School of Electronic Information and Communications, Huazhong University of Science and Technology, Wuhan 430074, China*
[3]*Institute of Artificial Intelligence, Chinese Academy of Sciences, Shenzhen 518000, China*

Received 23 September 2024/Revised 7 February 2025/Accepted 2 April 2025/Published online 15 April 2026

Abstract Temporal motion modeling has always been a key component in multiple object tracking (MOT), which can ensure smooth trajectories and provide accurate positional information to enhance association precision. However, current motion models struggle to be both efficient and effective across different application scenarios. To this end, we propose TrackSSM, inspired by the recently popular state space models (SSMs), a unified encoder-decoder motion framework that uses a data-dependent state space model to perform temporal motion prediction of trajectories. Specifically, we propose Flow-SSM, a module that utilizes the position and motion information from historical trajectories to guide the temporal state transition of object bounding boxes. Based on Flow-SSM, we design a flow decoder. It is composed of a cascaded motion decoding module employing Flow-SSM, which can use the encoded flow information to complete the temporal position prediction of trajectories. Additionally, we propose a step-by-step linear (S^2L) training strategy. By performing linear interpolation between the positions of the object in the previous frame and the current frame, we construct the pseudo labels of step-by-step linear training, ensuring that the trajectory flow information can better guide the object bounding box in completing temporal transitions. TrackSSM utilizes a simple Mamba-Block to build a motion encoder for historical trajectories, forming a temporal motion model with an encoder-decoder structure in conjunction with the flow decoder. TrackSSM is applicable to various tracking scenarios and achieves excellent tracking performance across multiple benchmarks, further extending the potential of SSM-like temporal motion models in multi-object tracking tasks. Code and models are publicly available at https://github.com/Xavier-Lin/TrackSSM.

Keywords 2D multi-object tracking, state space model (SSM), temporal motion model, hidden state, flow information

Citation Hu B, Luo R, Liu Z L, et al. TrackSSM: a general motion predictor using a state space model. Sci China Inf Sci, 2026, 69(5): 152105, https://doi.org/10.1007/s11432-024-4849-2

1 Introduction

The modeling of linear and nonlinear motion has always been a key issue in multi-object tracking (MOT) tasks [1]. In scenarios with intense motion, such as dance scenes [2], sports [3], and autonomous driving [4], robust and efficient motion modeling has become an essential component of high-performance trackers. Although previous trackers [5–10] have achieved advanced performance on multiple benchmarks, robust and efficient motion modeling for various scenarios remains a significant challenge.

Successful motion modeling needs to ensure the following two points: (1) robustness to diverse motion patterns; (2) high inference efficiency. The current mainstream motion models in MOT adopt the Kalman filter [11], which is based on the assumption of constant velocity and is data-independent. It typically uses an equation of constant velocity motion to compute the prior state of a trajectory and update this state with matched observations to predict the trajectory position at the next time step. However, when the actual movement of a target significantly deviates from the motion prior, it can lead to erroneous trajectory associations. Some approaches [12–15] use attention-based autoregressive methods for temporal propagation of trajectories and demonstrate superior performance in nonlinear motion scenarios. With the increase in the number of tracking targets, attention-based autoregressive modeling leads to a quadratic growth in computational cost. Additionally, some methods [16, 17] employ convolutional neural networks (CNNs) for temporal autoregressive modeling, integrating them with detection networks within the same framework to form Siamese or shared-parameter networks. Although such approaches improve computational efficiency, they may lead to feature conflicts between tracking and detection tasks, resulting in weaker detection performance.

* Corresponding author (email: liuwy@hust.edu.cn)

© Science China Press 2026

Recently, state space models (SSMs) [18, 19] have achieved widespread success in efficiently handling long sequence tasks, owing to their effective computation of sequential information and state transition modeling. Inspired by SSM, we propose a unified motion framework based on data-dependent SSM, named TrackSSM. It follows an encoder-decoder architecture. The encoder is composed of stacked naive Mamba [20] modules, which aggregate the position and motion representations of historical trajectories to obtain the trajectory flow information. The decoder consists of cascaded motion decoding modules from our proposed Flow-SSM, which utilize the flow information obtained from the encoder to guide the temporal position prediction of the current frame trajectories. Additionally, to improve the accuracy of trajectory position prediction, we propose a step-by-step linear (S^2L) training strategy. By linearly interpolating the trajectory positions between the current frame and the previous frame, we construct step-by-step linear training pseudo labels, guiding the bounding box to complete temporal transitions in a progressive linear manner. Compared to Mamba, we parameterize the SSM using the flow information encoded from historical trajectories, resulting in Flow-SSM. It effectively handles various linear and nonlinear motion target position transitions.

Benefiting from the efficient computation of the Mamba module, the inference speed of TrackSSM with the YOLOX-L [21] detector can reach up to 27.5 FPS, surpassing most attention-based temporal autoregressive motion models [8, 10, 12, 14, 15]. With a fixed detector model and hyper-parameter configuration, the TrackSSM with the YOLOX-X [21] detector achieves comparable performance to the baseline ByteTrack [7], which employs the Kalman filter (KF) [11] as the motion model, on the MOT17 [22] test set. On the DanceTrack [2] test set, ByteTrack integrated with TrackSSM achieves a tracking performance of 57.7 HOTA [23], representing a gain of 10.9 HOTA compared to the baseline. On the SportsMOT [3] test set, ByteTrack with TrackSSM achieves a tracking performance of 74.4 HOTA, a gain of 11.0 HOTA compared to the baseline. Notably, TrackSSM paired with the detector can infer at real-time speed while incurring less computational overhead. Experimental results across different benchmarks demonstrate that TrackSSM has the potential to become a universal motion framework for multi-object tracking tasks.

Our contributions are summarized as follows.

• We propose Flow-SSM, which leverages the flow information from historical frame trajectories to perform the temporal state transitions of trajectory boxes.

• Based on Flow-SSM, we design the flow decoder, which performs step-by-step refinement of trajectory boxes output by each decoder layer by passing trajectory boxes and hidden states between decoder layers.

• We propose a S^2L training strategy. By performing linear interpolation on trajectory positions and constructing step-by-step linear training pseudo labels, we ensure that the flow information from historical frame trajectories can more accurately guide the object bounding box in performing temporal predictions.

• Combining the above designs, we propose TrackSSM, a simple and effective motion model with the encoder-decoder structure. TrackSSM is applicable to various tracking scenarios and achieves excellent performance across multiple tracking benchmarks.

2 Related work

2.1 2D multiple object tracking

Current mainstream 2D multi-object tracking methods can be categorized into tracking-by-detection (TBD) paradigms and joint detection and tracking (JDT) paradigms. Most TBD methods [5–7, 24–33] typically use KF as the motion model, which predicts the prior position of the trajectory at the next time step and associates it with the corresponding detection. Although TBD methods have achieved impressive performance on multiple tracking benchmarks, their performance is somewhat limited by hyper-parameters and specific scenarios. To compensate for the shortcomings of the KF motion model in modeling complex scenarios, appearance features are introduced as an important association metric. These features help recall lost trajectories when occlusion and loss occur. While the powerful appearance models [34–37] are beneficial for accurate tracking, the efficiency of the tracker decreases as the number of objects in the scene increases. To improve the robustness of trackers across different scenarios and reduce the number of hyper-parameters, JDT methods have been proposed to simultaneously perform object detection and trajectory temporal position prediction tasks. CenterTrack [16] tracks objects as points, detecting and tracking the center points of objects while predicting their temporal displacements. SiamMOT [17] uses a Siamese network to jointly optimize the detection and tracking tasks within the same framework, performing temporal regression of trajectory positions via the tracking network. TransTrack [12] is the first to introduce the Transformer architecture into tracking algorithms, constructing a tracking method dependent on track queries. It represents trajectories

as queries to predict the position of the previous frame trajectory in the current frame. TrackFormer [14] is the first to propose a tracking method based on continuous temporal autoregression of trajectory queries, allowing trajectory queries to possess true continuous time tracking capabilities. MOTR [15] represents both detection and tracking tasks as a set prediction problem within a single stage, achieving truly end-to-end multi-object tracking. MeMOTR [10] builds upon MOTR via the introduction of long-term memory injection with customized memory-augmented attention layers, delivering more stable and discriminative trajectory embeddings for identical objects, thereby significantly enhancing the model's target association capability.

2.2 State space models

SSMs [18, 19] generally refer to a class of models that utilize hidden states for sequential autoregression of objects. The current widely used state space model is S4 [18], whose autoregressive process can be described as follows:

$$\begin{aligned} h_t &= \overline{\mathbf{A}} h_{t-1} + \overline{\mathbf{B}} x_t, \\ y_t &= \mathbf{C} h_t + \mathbf{D} x_t, \end{aligned} \tag{1}$$

where x_t represents the input signal, h_t is the hidden state, and y_t is the output signal. $\overline{\mathbf{A}}$, $\overline{\mathbf{B}}$, $\mathbf{C}$ and $\mathbf{D}$ are four matrix parameters, with $\overline{\mathbf{A}}$ and $\overline{\mathbf{B}}$ being discrete parameters that can be obtained through zero-order hold (ZOH) and Euler discretization rules. Although S4 offers a concise and efficient solution for long-sequence modeling, further exploration is needed for modeling longer and more complex sequences. S5 [19] introduces efficient parallel scanning and MIMO-SSM into the S4 layer, further enhancing the sequence modeling capabilities of SSM. Recently, Gu and Dao [20] introduced a data-driven SSM layer, referred to as the Mamba module. The core design of the Mamba module involves parameterizing the $\mathbf{\Delta}$, $\mathbf{B}$ and $\mathbf{C}$ matrix parameters using representations extracted from sequence data. Benefiting from its data-driven nature, the Mamba module is better equipped to capture sequence features and enhance long-sequence modeling capabilities. Notably, the Mamba module scales linearly with sequence length, while maintaining low memory overhead and high inference efficiency. In this work, we design a motion prediction model based on the SSM structure and thoroughly explore the potential of the SSM architecture in temporal prediction tasks.

2.3 Methods for motion modeling in MOT

Motion modeling in multi-object tracking can be divided into two categories: heuristic motion models and learnable motion models. Heuristic methods typically use fixed motion priors and a set of hyper-parameters to control the trajectory motion process, with the KF [11] being a typical example. While KF motion models have been successful in most tracking benchmarks [22, 38, 39], they can lead to failed tracking results in benchmarks with more intense motion. To address the limitations of the KF, GIAOTracker [40] proposed the NSA Kalman filter motion model, which aims to adaptively adjust the noise scale based on the quality of object detection, achieving success in multi-object tracking benchmarks that involve complex motion. Other approaches [27, 29, 41] use camera motion compensation to mitigate the intensity of object motion. Both naive and NSA Kalman filters come with a large number of hyper-parameters, which poses a potential risk of being limited to specific types of scenarios. As a result, learnable motion models have gradually attracted researchers' attention, thanks to their data-driven nature. Tracktor [42] is the first tracker to propose a learnable motion model, using trajectory boxes as regions of interest (RoI) [43] in each frame, extracting corresponding RoI features to regress the trajectory boxes to the current frame. MotionTrack [44] learns the representation of the trajectory at historical moments and uses it to predict the movement of the trajectory at the next moment. Trajectory autoregressive models based on self-attention mechanisms [45,46] can partially overcome the challenge of motion modeling for occluded objects. However, they can cause conflicts between detection and tracking tasks during the tracking process, weakening detection performance. DiffMOT [47] constructs a temporal diffusion motion model to replace the KF, viewing the regression process of the trajectory box from the previous frame to the current frame as a diffusion-denoising [48] process, achieving some success across different tracking benchmarks. Nevertheless, efficient and accurate motion modeling remains an area that requires further exploration. Recently, MambaTrack [49], an SSM tracker, achieves comparable tracking performance in complex motion scenarios. It regards the bi-directional Vision Mamba [50] encoder as the motion module, which can predict the location of trajectories per time step. However, MambaTrack has not achieved real-time inference speed, which limits its application in real-time scenarios. In this paper, we further explore the capability of SSM-like approaches in fitting trajectory motion patterns, improving both accuracy and efficiency in multi-object tracking.

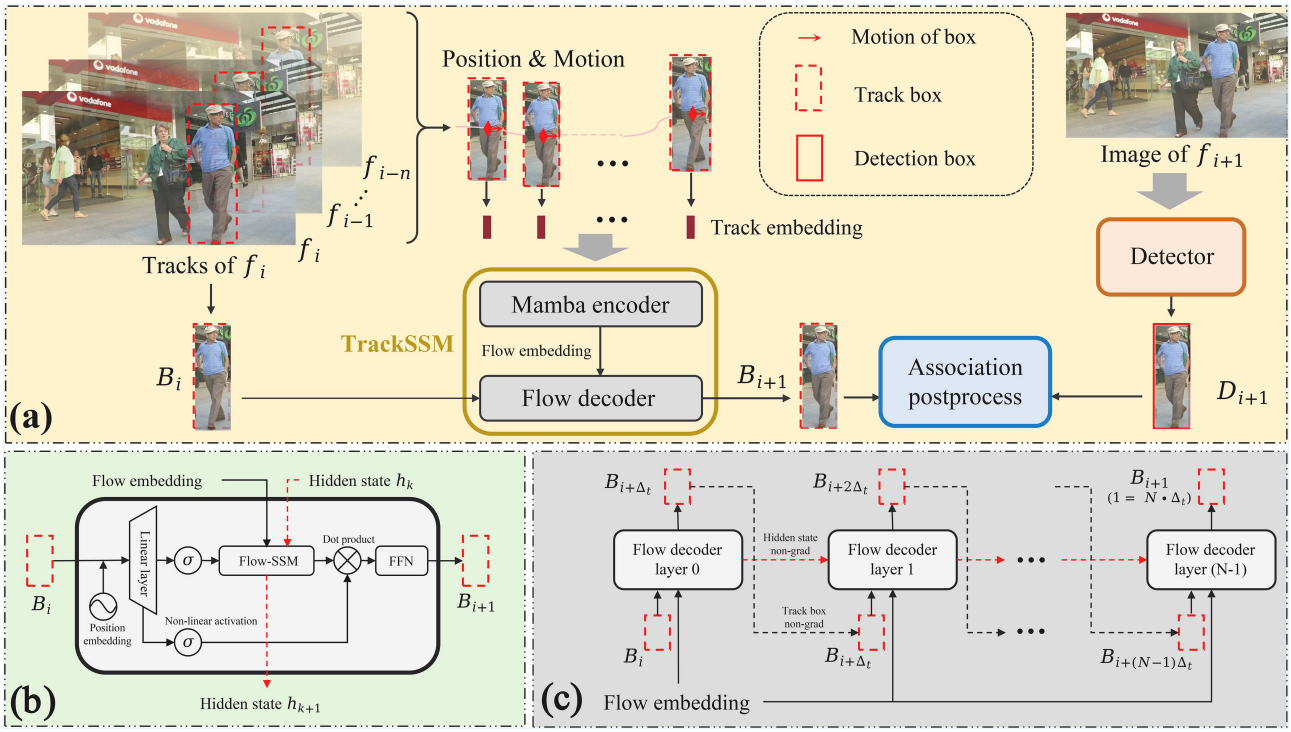


Figure 1 (Color online) The illustration of the overall framework with the TrackSSM motion model. (a) The tracking framework is integrated with the TrackSSM motion model. TrackSSM consists of the Mamba encoder [20] and the flow decoder, which enables motion prediction for trajectories. (b) The internal structure of a single decoder layer, which includes the Flow-SSM module we proposed. (c) The network structure of the flow decoder is composed of several cascaded decoder layers.

3 Method

3.1 General framework

The overall tracking framework via the TrackSSM motion model is shown in Figure 1(a). Given the position and motion information of a trajectory $\{T_{i-k} = (x_c, y_c, w, h, \Delta_x, \Delta_y, \Delta_w, \Delta_h)\}_{k=n}^{0}$ for n historical frames, we encode the trajectory information T_{i-k} at each time step into trajectory embeddings $\mathcal{T}_{i-k} \in \mathbf{R}^m$, forming a sequence of trajectory embeddings $\{\mathcal{T}_{i-k}\}_{k=n}^{0}$. The embedding sequence is then fed into a naive Mamba encoder [20], with the output representation at the final time step serving as the motion flow information of the trajectories, which we refer to as the flow feature $\mathcal{F} \in \mathbf{R}^m$. The flow feature contains abundant historical information about the trajectory with position and motion. Subsequently, we use the flow feature $\mathcal{F}$ as guidance, feeding it into a designed flow decoder to guide the trajectory box B_i in predicting its position B_{i+1}, which can obtain a prediction track box at the time $(i+1)$. During the tracking phase, the trajectory prediction box B_{i+1} is associated with the detection box D_{i+1} obtained by the detector, and the association process is similar to that in ByteTrack [7].

3.2 Flow-SSM

To achieve the process of using flow features to guide trajectory boxes for temporal prediction, we design Flow-SSM. The algorithm pseudo-code is shown in Algorithm 1, where `B` represents the batch size, `D` denotes the dimension of the state space model, and `N` is the state dimension. We adopt the ZOH method to discretize the parameters $\mathbf{A}$, $\mathbf{B}$ into $\overline{\mathbf{A}}$, $\overline{\mathbf{B}}$ through the time scale $\boldsymbol{\Delta}$, following the method in [20]. Compared to the Mamba module, Flow-SSM parameterizes the $\boldsymbol{\Delta}$, $\mathbf{B}$ and $\mathbf{C}$ matrices using flow features $\mathcal{F}$ to achieve temporal guidance of the trajectory embeddings $\mathcal{E}_i$. Unlike traditional state space models [18–20], Flow-SSM does not perform sequence modeling but instead predicts the state of the input signal at the next time step, similar to the Kalman filter.

3.3 Flow decoder

To accurately regress the trajectory box from the previous frame to the current frame, we design a cascaded motion decoder, referred to as the flow decoder. The flow decoder is composed of N identical decoder layers cascaded together, with each decoder layer containing a Flow-SSM module. The specific structure of the decoder layer is

Algorithm 1 Flow-SSM.

Input: Track position embedding $\mathcal{E}_i : (\mathtt{B}, \mathtt{D})$; flow features $\mathcal{F} : (\mathtt{B}, \mathtt{M})$; hidden state $h : (\mathtt{B}, \mathtt{D}, \mathtt{N})$.
Output: Track position embedding $\mathcal{E}_{i+1} : (\mathtt{B}, \mathtt{D})$; hidden state $h' : (\mathtt{B}, \mathtt{D}, \mathtt{N})$.
/* Parameterize data-independent matrices */
$\mathbf{A} : (\mathtt{D}, \mathtt{N}) \leftarrow$ Parameter;
$\mathbf{D} : (\mathtt{D},) \leftarrow$ Parameter;
/* Parameterize data-dependent matrices via flow features */
$\mathbf{\Delta} : (\mathtt{B}, \mathtt{D}), \mathbf{B} : (\mathtt{B}, \mathtt{N}), \mathbf{C} : (\mathtt{B}, \mathtt{N}) \leftarrow \mathbf{Linear}(\mathcal{F})$;
/* Discretize */
$\overline{\mathbf{A}} : (\mathtt{B}, \mathtt{D}, \mathtt{N}) \leftarrow \mathbf{Exp}(\mathbf{\Delta} \bigotimes \mathbf{A})$;
$\overline{\mathbf{B}} : (\mathtt{B}, \mathtt{D}, \mathtt{N}) \leftarrow \mathbf{\Delta} \bigotimes \mathbf{B}$;
/* Running SSM */
$\mathcal{E}_{i+1} : (\mathtt{B}, \mathtt{D}), h' : (\mathtt{B}, \mathtt{D}, \mathtt{N}) \leftarrow \mathbf{SSM}(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \mathbf{C}, \mathbf{D})(h, \mathcal{E}_i)$;
Return: $\mathcal{E}_{i+1}, h'$.

shown in Figure 1(b). Given a trajectory box B_i, we apply cosine positional encoding to transform it into a high-dimensional trajectory position embedding $\mathbf{E}_i$. Subsequently, we pass the trajectory position embedding $\mathbf{E}_i$ into a linear layer and split it along the dimension into two position features, $\mathcal{E}_i$ and $\mathcal{R}_i$. The position feature $\mathcal{E}_i$ is fed into the Flow-SSM module, resulting in the trajectory prediction feature $\mathcal{E}_{i+1}$. The feature $\mathcal{R}_i$ serves as a residual component, passing through a nonlinear activation layer, and is then multiplied by the trajectory prediction feature $\mathcal{E}_{i+1}$, enriching the representation of $\mathcal{E}_{i+1}$. Finally, $\mathcal{E}_{i+1}$ is passed through a feed-forward network (FFN) [46] to obtain the trajectory prediction box B_{i+1}.

However, the trajectory prediction B_{i+1} output by a single decoder layer is insufficient for achieving precise trajectory prediction. Therefore, we cascade N identical decoder layers to construct the flow decoder, with the overall framework shown in Figure 1(c). The flow embeddings obtained from the Mamba encoder [20] are applied to each decoder layer. The output trajectory box from the previous decoder layer serves as the input for the next decoder layer, allowing for precise refinement of the trajectory box position over time. Additionally, the hidden state acts as a messenger, transmitting the state of the trajectory box between the cascaded decoder layers, enabling the trajectory box to gradually regress according to the ground truth (GT) labels. The specific details of the regression process will be described in Subsection 3.4.

3.4 Step-by-step linear training strategy

The flow decoder refines the trajectory box through a cascading process. In each flow decoder layer, the refinement of the trajectory box is guided by the flow features. Based on the intuition that the flow features have the same guiding effect across all decoder layers, we propose an S^2L training strategy. The core of S^2L is to linearly decompose the temporal autoregressive process of the trajectory box into N simple regression steps. Specifically, given the trajectory box B_i at time i, the flow decoder regresses B_i to B_{i+1}. If there are N flow decoder layers, the regression process from B_i to B_{i+1} can be linearly decomposed into N sub-processes. It can be expressed as

$$\begin{aligned} \Delta_t &= \frac{(i+1)-i}{N}, \\ \{B_{i+(k+1)\Delta_t} \leftarrow B_{i+k\Delta_t}\}_{k=0}^{N-1} &= I(B_{i+1} \leftarrow B_i), \end{aligned} \tag{2}$$

where Δ_t is the time step, and I is the linear interpolation function. By performing linear interpolation between B_i and B_{i+1}, we can construct pseudo labels $\{B_{i+k\Delta_t}\}_{k=1}^{N}$ for supervising each flow decoder layer. From the perspective of the entire regression process, by constructing pseudo-labels obtained through linear interpolation, we encourage the flow decoder to learn the regression process from B_i to B_{i+1} in a linear recursive manner, which can be expressed as

$$B_i \xrightarrow{\mathbf{f}_1(h,\mathcal{F})} B_{i+\Delta_t} \xrightarrow{\mathbf{f}_2(h,\mathcal{F})} \cdots \xrightarrow{\mathbf{f}_N(h,\mathcal{F})} B_{i+N\Delta_t}, \tag{3}$$

where $\mathbf{f}$ represents each flow decoder layer, h is the hidden state, and $\mathcal{F}$ denotes the flow features. By using the step-by-step linear training strategy, the flow features guide the trajectory box through an equal amount of transformation in each decoder layer. This approach enables the flow decoder to handle more complex trajectory motions and improves the recall rate of lost trajectories.

3.5 Training loss of the TrackSSM

We use ground truth as well as pseudo labels obtained through linear interpolation to supervise TrackSSM. We employ the smooth L1 loss and generalized intersection over union (GIoU) [51, 52] loss for training TrackSSM,

specifically expressed as follows:

$$\mathcal{L}_{total} = \lambda_1 \mathcal{L}_{smoothL1} + \lambda_2 \mathcal{L}_{GIoU}, \tag{4}$$

where λ_1 and λ_2 are the weight coefficients for the smooth L1 loss and GIoU loss, respectively. The specific formula for the smooth L1 loss is as follows:

$$\mathcal{L}_{smoothL1} = \begin{cases} 0.5(\hat{\mathbf{B}} - \mathbf{B})^2, & |\hat{\mathbf{B}} - \mathbf{B}| < 1, \\ |\hat{\mathbf{B}} - \mathbf{B}| - 0.5, & \text{otherwise}, \end{cases} \tag{5}$$

where $\hat{\mathbf{B}}$ represents the trajectory box predicted by each decoder layer, and $\mathbf{B}$ represents the supervision labels. The matrix operations in (5) are element-wise.

4 Experiments

4.1 Setting

4.1.1 *Datasets*

We evaluate the performance of TrackSSM in pedestrian, dancing, and sports scenarios, corresponding to the MOT17 [22], DanceTrack [2], and SportsMOT [3] benchmarks, respectively. We merge the MOT17 and MOT20 [39] training sets, referring to the combined set as MIX. For reporting results on the MOT17 test set, we train TrackSSM on MIX. For reporting results on the DanceTrack and SportsMOT test sets, we train TrackSSM separately on their respective training sets. For the ablation experiments, we train TrackSSM on the MIX, DanceTrack and SportsMOT training sets, respectively, and perform ablation testing on the MOT17 training set, DanceTrack validation set, and SportsMOT validation set.

4.1.2 *Metrics*

We use the standard CLEAR metrics (MOTA, etc.) [53], HOTA [23], AssA, DetA and IDF1 [54] to comprehensively evaluate tracking performance. HOTA is an important metric for assessing the overall performance of detection and association. IDF1 is used to measure the precision and recall of trajectory associations. AssA and DetA specifically focus on measuring association accuracy and detection accuracy, respectively. Additionally, we use frames per second (FPS) to assess the efficiency of the tracker.

4.1.3 *Implementation details*

Training. During the training phase of TrackSSM, we train using trajectory segments rather than images, following the approach used in DiffMOT [47]. We select the position and motion information of historical frame trajectories with a time length of 5 as the input to TrackSSM. For the training setup, we set the default batch size to 2048 and use the Adam optimizer with a learning rate of 0.0001. The number of layers in the flow decoder is set to 6 by default. For the MIX, we train TrackSSM for a total of 160 epochs. For the DanceTrack [2], we train TrackSSM for 120 epochs. For the SportsMOT [3], we train TrackSSM for 340 epochs. Since some ground-truth boxes in the MOT17 [22] exceed the image boundaries, we omit the bounding box normalization step when training on the MIX dataset. Additionally, when training TrackSSM on the MIX and DanceTrack, we use only the smooth L1 loss. However, when training TrackSSM on the SportsMOT, we use both the smooth L1 loss and GIoU [52] loss. This is because the object displacement distances between adjacent frames in the SportsMOT are greater, and using the GIoU loss helps the motion model converge more quickly.

Inference. During the inference phase, we set the default resolution of the input image to 800×1440 and use the publicly available YOLOX-X [21] detector to infer detection results. To ensure a fair comparison with the baseline [7], we fix all hyper-parameter settings during tracking inference. The high-score detection threshold and low-score detection threshold are set to 0.6 and 0.1, respectively. For the non-maximum suppression (NMS) [43] post-processing, we fix the intersection over union (IoU) threshold at 0.7 and the confidence threshold at 0.01. The tracker uses positional information for association, without the involvement of appearance features.

Device. We train TrackSSM with 2 GeForce RTX 3090 GPUs. During the inference phase, we perform tracking using a single GeForce RTX 3090 GPU.

Table 1 The comparison of ByteSSM with other methods on the MOT17 test set. The $*$ indicates that this method is identical to ByteSSM in all settings except for the motion model. The best results are shown in bold.

Tracker	HOTA↑	MOTA↑	IDF1↑	AssA↑	DetA↑
Kalman filter					
FairMOT [5]	59.3	73.7	72.3	58.0	60.9
ByteTrack [7]	63.1	80.3	77.3	62.0	64.5
ByteTrack* [7]	62.8	78.7	76.8	62.1	63.8
OC-SORT [55]	63.2	78.0	77.5	63.4	63.2
SparseTrack [29]	65.1	81.0	80.1	65.1	65.3
Learnable motion					
CenterTrack [16]	52.2	67.8	64.7	51.0	53.8
TraDes [56]	52.7	69.1	63.9	50.8	55.2
QuasiDense [57]	53.9	68.7	66.3	52.7	55.6
TransTrack [12]	54.1	75.2	63.5	47.9	61.6
TransCenter [13]	54.5	73.2	62.2	49.7	60.1
TrackFormer [14]	57.3	74.1	68.0	54.1	60.9
MeMOTR [10]	58.8	72.8	71.5	58.4	59.6
GTR [58]	59.1	75.3	71.5	57.0	61.6
MOTR [15]	57.8	73.4	68.6	55.7	60.3
STDFormer [59]	60.9	78.4	73.1	58.4	–
MambaTrack+ [60]	61.1	78.1	73.9	–	–
ByteSSM	**61.4**	**78.5**	**74.1**	**59.6**	**63.6**

4.2 Evaluation of different benchmarks

We replace the Kalman filter [11] in ByteTrack [7] with the TrackSSM motion model. For ease of reference, we temporarily refer to ByteTrack using TrackSSM as ByteSSM. We evaluate the ByteSSM on the MOT17 [22], DanceTrack [2] and SportsMOT [3] benchmarks to compare with other methods and present the results as follows.

MOT17. The MOT17 [22] dataset contains frequent occlusions and slight camera movements, which pose a challenge to the ability of motion models to fit trajectories. Our method is compared with previous advanced methods [5,7,10,12–16,29,55–60] in Table 1. By using TrackSSM as the motion model, ByteSSM achieves the highest level among many learnable motion methods. Compared to the baseline ByteTrack [7], ByteSSM's performance is slightly weaker. This is because the constant velocity motion prior in the Kalman filter [11] naturally aligns well with the pedestrian movements in the MOT dataset [22, 38, 39].

DanceTrack. DanceTrack [2] is a more challenging benchmark for multi-object tracking with intense nonlinear motion, frequent occlusions, and similar appearances among objects. Our method is compared with previous advanced methods [5,7,12,15,16,27,29,47,49,55–58,60,61] in Table 2. With the same detection and hyper-parameter settings, ByteSSM that adopts the TrackSSM motion model achieves the association gain of 10.9 HOTA [23], 10.1 AssA and 5.7 IDF1 [54] compared to the baseline method. Among various trackers with learnable motion modules, ByteSSM achieves the best tracking performance. This demonstrates TrackSSM's exceptional ability to model nonlinear motion trajectories.

SportsMOT. SportsMOT [3] is a large-scale multi-object tracking benchmark designed for sports scenarios. It includes three types of sports scenes: soccer, basketball, and volleyball. Compared to the MOT challenge benchmark [22, 38, 39], SportsMOT presents diverse motion patterns and complex nonlinear movements, posing a significant challenge to the tracker's ability to model complex trajectory motions. Our method is compared with previous advanced methods [3,5,7,10,12,16,44,47,49,55,57,58,60,61] in Table 3. With the same detection and hyper-parameter settings, ByteSSM achieves the association gain of 11.0 HOTA, 11.1 AssA and 4.2 IDF1 compared to the baseline. Among various tracking methods utilizing learnable motion modules, ByteSSM continues to maintain the best tracking performance. This strongly demonstrates the potential of TrackSSM as a universal motion predictor.

4.3 Ablation studies

4.3.1 *The impact of different motion models on tracking performance*

By keeping the detection and hyper-parameter settings fixed, we replace the Kalman filter [11] motion module in ByteTrack [7] with TrackSSM, DiffMOT [47] and global motion compensation (GMC) [27] in succession to observe the performance differences between different motion models. We conduct inference testing using the aforementioned

Table 2 The comparison of ByteSSM with other methods on the DanceTrack test set. The $*$ indicates that this method is identical to ByteSSM in all settings except for the motion model. The best results are shown in bold.

Tracker	HOTA↑	MOTA↑	IDF1↑	AssA↑	DetA↑
Kalman filter					
FairMOT [5]	39.7	82.2	40.8	23.8	66.7
ByteTrack* [7]	46.8	89.6	51.8	30.9	71.3
ByteTrack [7]	47.7	89.6	53.9	32.1	71.0
BoT-SORT [27]	54.7	91.3	56.0	37.8	79.6
OC-SORT [55]	55.1	92.0	54.6	38.3	80.3
SparseTrack [29]	55.7	91.3	58.1	39.3	79.2
Learnable motion					
CenterTrack [16]	41.8	86.8	35.7	22.6	78.1
TraDes [56]	43.3	86.2	41.2	25.4	74.5
TransTrack [12]	45.5	88.4	45.2	27.5	75.9
GTR [58]	48.0	84.7	50.3	31.9	72.5
QuasiDense [57]	54.2	87.7	50.4	36.8	80.1
MOTR [15]	54.2	79.7	51.5	40.2	73.5
DiffMOT* [47]	56.1	92.2	55.7	38.5	**81.9**
MambaTrack+ [60]	56.1	90.3	54.9	39.0	80.8
ETTrack [61]	56.4	92.2	57.5	39.1	81.7
MambaTrack [49]	56.8	90.1	**57.8**	39.8	80.1
ByteSSM	**57.7**	**92.2**	57.5	**41.0**	81.5

Table 3 The comparison of ByteSSM with other methods on the SportsMOT test set. The $*$ indicates that this method is identical to ByteSSM in all settings except for the motion model. The best results are shown in bold.

Tracker	HOTA↑	MOTA↑	IDF1↑	AssA↑	DetA↑
Kalman filter					
FairMOT [5]	49.3	86.4	53.5	34.7	70.2
ByteTrack [7]	64.1	95.9	71.4	52.3	78.5
ByteTrack* [7]	63.4	95.7	70.3	51.3	78.4
OC-SORT [55]	73.7	96.5	74.0	61.5	88.5
Learnable motion					
GTR [58]	54.5	67.9	55.8	45.9	64.8
QuasiDense [57]	60.4	90.1	62.3	47.2	77.5
CenterTrack [16]	62.7	90.8	60.0	48.0	82.1
TransTrack [12]	68.9	92.6	71.5	57.5	82.7
MeMOTR [10]	70.0	91.5	71.4	59.1	83.1
MambaTrack+ [60]	71.3	94.9	71.1	58.6	86.7
MambaTrack [49]	72.6	95.3	72.8	60.3	87.6
DiffMOT* [47]	74.0	96.8	73.7	61.7	**88.9**
MotionTrack [44]	74.0	96.6	74.0	61.7	88.8
MixSort-OC [3]	74.1	96.5	74.4	62.0	88.5
ETTrack [61]	74.3	96.8	74.5	62.1	88.8
ByteSSM	**74.4**	**96.8**	**74.5**	**62.4**	88.8

motion modules on the MOT17 [22] training set, DanceTrack [2] validation set and SportsMOT [3] validation set. The results are shown in Table 4.

In pedestrian scenarios, the Kalman filter with a constant velocity motion prior achieves better tracking results. This is because most pedestrian objects move at approximately constant speeds. However, in dancing and sports scenarios, the motion of the objects becomes nonlinear and the targets undergo significant deformations. In these cases, the constant velocity assumption no longer provides a reliable positional prior, leading to weaker tracking performance. Compared to the Kalman filter (without GMC), TrackSSM achieves comparable performance in pedestrian scenarios and significantly outperforms the Kalman filter motion model in dancing and sports scenarios. This indicates that learnable motion models can better fit the nonlinear and non-rigid motions of trajectories, thereby improving association accuracy during tracking. Moreover, when compared to DiffMOT, which is also a learnable motion model, TrackSSM consistently outperforms DiffMOT across all three scenarios. This demonstrates

Table 4 Comparison of the tracking performance with different motion modules. The best results are shown in bold.

Motion model	MOT17			DanceTrack			SportsMOT		
	HOTA↑	AssA↑	IDF1↑	HOTA↑	AssA↑	IDF1↑	HOTA↑	AssA↑	IDF1↑
Kalman filter									
KF [11]	75.0	72.4	83.3	47.1	31.4	51.0	67.8	57.2	76.0
KF+OC-SORT [55]	74.1	71.8	81.7	52.3	35.4	51.9	77.7	65.6	76.6
KF+GMC [27]	**77.1**	**75.1**	**85.5**	53.4	36.5	**53.9**	79.8	69.6	80.8
Learnable motion									
DiffMOT [47]	73.9	69.6	81.3	53.7	36.7	53.3	80.9	70.1	80.3
TrackSSM	74.9	71.4	81.7	**53.8**	**36.8**	53.7	**81.2**	**70.7**	**80.8**

Table 5 Comparison of the tracking performance with different lengths of trajectory segments. The best results are shown in bold.

Length	DanceTrack			SportsMOT		
	HOTA↑	AssA↑	IDF1↑	HOTA↑	AssA↑	IDF1↑
3	**53.9**	**36.9**	53.0	**81.5**	**71.2**	**81.2**
5	53.8	36.8	**53.7**	81.2	70.7	80.8
10	52.5	34.9	50.0	81.1	70.6	80.6
20	52.9	35.6	52.3	80.9	70.2	80.1
40	53.9	36.8	52.8	80.6	69.6	79.6

that TrackSSM can effectively handle nonlinear motion trajectories, proving its broad applicability across different scenarios.

4.3.2 *The impact of trajectory segment with different lengths on tracking performance*

We explore the impact of the length of trajectory segments input to TrackSSM on tracking performance. Specifically, during the training phase, we train trajectory segments with historical time lengths of 3, 5, 10, 20 and 40, separately, while keeping the training settings consistent. During the inference phase, we use trajectory information with historical time lengths of 3, 5, 10, 20 and 40 for motion prediction, separately, again maintaining consistent detection and hyper-parameter settings. The experimental results are shown in Table 5.

When the historical time length of trajectory segments used in both the training and inference phases is set to 3, ByteSSM achieves the best tracking performance. This suggests that trajectory information closer to the current time is more relevant to future trajectory predictions. For the DanceTrack [2] dataset, the tracking performance of ByteSSM continues to improve as the historical time length of the input trajectory segments increases. We speculate that the movement and actions of targets in dancing scenes often exhibit a certain degree of periodicity. As the historical trajectory information increases, TrackSSM may learn the complete motion cycle of the tracked target, which facilitates accurate prediction of the future position of dancers.

4.3.3 *The impact of different numbers of decoder layers on tracking performance*

We further explore the impact of the number of decoder layers on tracking performance. We fix all training settings and train TrackSSM with 1, 2, 3, 6 and 12 decoder layers, respectively. During the inference phase, we fix all detection and hyper-parameter settings and use TrackSSM with 1, 2, 3, 6, and 12 decoder layers for motion prediction, separately. The experimental results are shown in Table 6.

For the flow decoder with a single layer, the training process involves directly predicting the trajectory's future position without the need for a step-by-step linear prediction process. Despite this, the flow decoder with one layer still achieves decent tracking performance, indicating that even a single flow decoder layer can perform trajectory prediction with reasonable accuracy. When the number of decoder layers is set to 6, ByteSSM achieves the best tracking performance. Therefore, we select the flow decoder with 6 layers as the default configuration for TrackSSM.

4.3.4 *The impact of the step-by-step linear training strategy on tracking performance*

To observe the impact of the S^2L training strategy on TrackSSM's motion modeling, we conduct an ablation analysis for S^2L on the DanceTrack [2] and SportsMOT [3] validation datasets, separately. We perform training plans with and without the S^2L strategy, keeping all other configurations unchanged. The experimental results are shown in Table 7.

Table 6 Comparison of the tracking performance with different numbers of decoder layers. The best results are shown in bold.

Number of layers	DanceTrack			SportsMOT		
	HOTA↑	AssA↑	IDF1↑	HOTA↑	AssA↑	IDF1↑
1	53.4	36.3	52.8	80.3	69.2	79.4
2	52.8	35.3	50.8	80.0	68.8	79.0
3	52.4	34.8	51.2	79.6	68.0	78.6
6	**53.8**	**36.8**	**53.7**	**81.2**	**70.7**	**80.8**
12	52.8	35.2	51.0	80.6	69.7	80.1

Table 7 Comparison of the tracking performance with or without the step-by-step linear training strategy. The best results are shown in bold.

w/S^2L	DanceTrack			SportsMOT		
	HOTA↑	AssA↑	IDF1↑	HOTA↑	AssA↑	IDF1↑
	51.4	33.5	49.2	80.9	70.4	80.4
✓	**53.8**	**36.8**	**53.7**	**81.2**	**70.7**	**80.8**

Table 8 Comparison of tracking performance of TrackSSM using different detectors. The numbers in parentheses indicate the parameters of the proposed TrackSSM module.

TrackSSM	FPS	Parameters	HOTA
TrackSSM+YOLOX-X	20.3	104M (5.1M)	57.7
TrackSSM+YOLOX-L	27.5	59.2M (5.1M)	57.3
TrackSSM+YOLOX-M	29.8	30.4M (5.1M)	52.8
TrackSSM+YOLOX-S	31.9	14.1M (5.1M)	48.5

Clearly, after applying the S^2L strategy, the association performance of trackers improves across validation sets in both scenarios. On the DanceTrack validation set, TrackSSM trained with the S^2L strategy yields performance gains of 2.4 HOTA [23], 3.3 AssA, and 4.5 IDF1 [54]. This indicates that the S^2L strategy helps TrackSSM more accurately regress nonlinear motion trajectory boxes and recalls lost trajectories (as evidenced by the significant improvement in the IDF1 metric). On the SportsMOT validation set, TrackSSM trained with the S^2L strategy still provides the tracker with performance gains of 0.3 HOTA, 0.3 AssA, and 0.4 IDF1. This demonstrates the general applicability of the S^2L strategy across different scenarios. Notably, the S^2L strategy brings more substantial gains to the tracker on the DanceTrack dataset. We speculate that it is due to two factors: (1) the DanceTrack dataset contains more non-rigid motions; (2) the S^2L strategy focuses on handling non-rigid deformation motion of trajectory boxes, which is a type of nonlinear motion. By linearly decomposing non-rigid deformations into several simple, equal transformation steps, the S^2L strategy enables the flow decoder to more easily learn the non-rigid motion of trajectory boxes.

4.3.5 *The impact of different detectors on tracking performance*

We use publicly available different versions of the YOLOX [21] detector for inference on the DanceTrack [2] test set and employ TrackSSM with fixed weights for motion prediction. The experimental results are shown in Table 8. The parameter count of TrackSSM is indicated in parentheses.

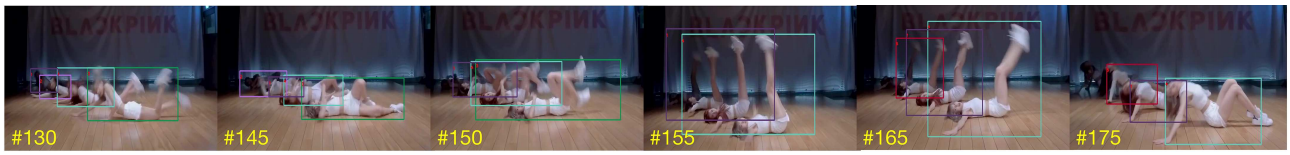
When a more lightweight detector is used, the accuracy of the tracker decreases, which aligns with the common pattern observed in TBD paradigms [24]. It is worth noting that the TrackSSM model has only 5.1M parameters, indicating its potential for deployment on edge devices. Additionally, when using the YOLOX-L detector, ByteSSM achieves a great trade-off between efficiency and tracking accuracy, with a running speed of 27.5 FPS, enabling real-time multi-object tracking.

4.4 Visualizations

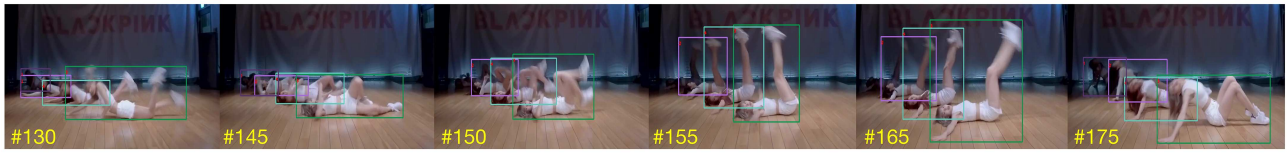
To exhibit the motion modeling capability of TrackSSM intuitively, we conduct several different visualization experiments, as shown in Figures 2–4, respectively.

We separately visualize the inference results of TrackSSM and the baseline on the same video clips under consistent inference settings, as shown in Figure 2. Compared to the baseline method, TrackSSM effectively handles nonlinear and abrupt motion in trajectories.

We visualize a set of tracking results with and without the S^2L training strategy with consistent training settings on the same video clip, as shown in Figure 3. The results show that TrackSSM with the S^2L training strategy is



(a) The visualization of ByteTrack on DanceTrack validation set

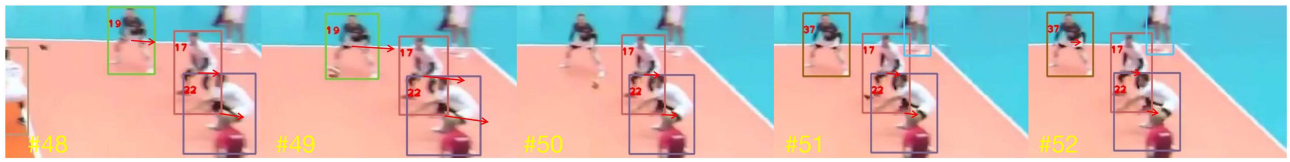


(b) The visualization of TrackSSM on DanceTrack validation set

(c) The visualization of ByteTrack on SportsMOT validation set

(d) The visualization of TrackSSM on SportsMOT validation set

Figure 2 (Color online) The visualization of TrackSSM and baseline methods on the same video clip, where the yellow numbers indicate the frame numbers of the images in the video.



(a) The visualization of TrackSSM with S^2L on SportsMOT validation set

(b) The visualization of TrackSSM without S^2L on SportsMOT validation set

Figure 3 (Color online) The visualization comparison of TrackSSM with and without the S^2L training strategy. We perform inference using TrackSSM on the same video clip. The yellow numbers indicate the frame index of the corresponding image within the video clip. Red arrows represent the displacement of the target bounding box center relative to the previous frame. We select video clips with intense and abrupt movements for visualization to validate the robustness of the S^2L strategy with sudden motion changes.

Figure 4 (Color online) The case where TrackSSM fails to associate targets in crowded and frequently occluded scenarios. The trajectories of occluded targets all experience ID switches, indicating that TrackSSM is unable to effectively model the motion of occluded trajectories.

better at handling abrupt trajectory movements.

Additionally, we investigate the motion modeling capability of TrackSSM for occluded trajectories, with the corresponding visualizations presented in Figure 4. TrackSSM exhibits weaker motion fitting performance for occluded trajectories, particularly in scenarios involving crowded environments with frequent occlusions.

4.5 Limitations

Although TrackSSM achieves good performance across multiple object tracking benchmarks in different scenarios, it still struggles to model the motion of long-term occluded trajectories. Similar to OC-SORT, TrackSSM relies entirely on observations for trajectory updates. For lost trajectories, TrackSSM can only perform trajectory prediction without correcting the trajectory state values. This leads to the accumulation of motion prediction errors over successive time steps. Furthermore, TrackSSM is merely a motion predictor and does not utilize target appearance representations. Developing a general spatiotemporal motion model that integrates both positional and appearance cues will be a promising direction for our future research.

5 Conclusion

We propose a simple and efficient motion model with an encoder-decoder structure, named TrackSSM. It uses a naive Mamba module to build the encoder, which converts the position and motion information of historical trajectories into flow features. In the decoding phase, to enhance nonlinear motion fitting, we introduce Flow-SSM. It uses flow features as a guide, facilitating precise temporal autoregression of the trajectory boxes. To further improve the accuracy of trajectory prediction, we carefully design the flow decoder, which is composed of several identical decoder layers that are cascaded together to refine the trajectory boxes step by step. Additionally, we propose an S^2L training strategy, which linearly decomposes the regression process of the trajectory boxes into several simple transformation steps. This strategy enhances TrackSSM's ability to model the motion of lost and complex trajectories. Compared to the popular Kalman filter [11] motion model, TrackSSM adapts to object motion in various scenarios and provides precise trajectory predictions for trackers. When compared to motion models using attention mechanisms [8, 12–15, 62], TrackSSM achieves significant motion prediction capabilities with much lower computational overhead, demonstrating its efficiency and robustness. In the future, we will continue to explore the potential of SSM-like tracking models in both the spatial-temporal dimensions, not just the temporal dimension. We also hope that this work will inspire the design of SSM-based decoder structures and anticipate the development of more elegant methods.

References

1 Vandenhende S, Georgoulis S, Van Gansbeke W, et al. Multi-task learning for dense prediction tasks: a survey. IEEE Trans Pattern Anal Mach Intell, 2022, 44: 3614–3633
2 Sun P, Cao J, Jiang Y, et al. Dancetrack: multi-object tracking in uniform appearance and diverse motion. ArXiv:2111.14690
3 Cui Y, Zeng C, Zhao X, et al. Sportsmot: a large multi-object tracking dataset in multiple sports scenes. ArXiv:2304.05170
4 Yu F, Chen H, Wang X, et al. Bdd100k: a diverse driving dataset for heterogeneous multitask learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020. 2636–2645
5 Zhang Y, Wang C, Wang X, et al. FairMOT: on the fairness of detection and re-identification in multiple object tracking. Int J Comput Vis, 2021, 129: 3069–3087
6 Wang Z, Zheng L, Liu Y, et al. Towards real-time multi-object tracking. In: Proceedings of the European Conference on Computer Vision (ECCV), 2020
7 Zhang Y, Sun P, Jiang Y, et al. Bytetrack: multi-object tracking by associating every detection box. In: Proceedings of the European Conference on Computer Vision, 2022
8 Zhang Y, Wang T, Zhang X. Motrv2: bootstrapping end-to-end multi-object tracking by pretrained object detectors. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023
9 Zhao Z, Wu Z, Zhuang Y, et al. Tracking objects as pixel-wise distributions. In: Proceedings of European Conference on Computer Vision, 2022
10 Gao R, Wang L. MeMOTR: Long-term memory-augmented transformer for multi-object tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2023. 9901–9910
11 RE K. A new approach to linear filtering and prediction problems. J Fluids Eng, 1960, 82: 35–45
12 Sun P, Jiang Y, Zhang R, et al. Transtrack: multiple-object tracking with transformer. ArXiv:2012.15460
13 Xu Y, Ban Y, Delorme G, et al. Transcenter: transformers with dense queries for multiple-object tracking. ArXiv:2103.15145
14 Meinhardt T, Kirillov A, Leal-Taixe L, et al. Trackformer: multi-object tracking with transformers. In: Proceedings of The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2022
15 Zeng F, Dong B, Zhang Y, et al. Motr: end-to-end multiple-object tracking with transformer. In: Proceedings of European Conference on Computer Vision (ECCV), 2022
16 Zhou X, Koltun V, Krähenbühl P. Tracking objects as points. In: Proceedings of European Conference on Computer Vision, 2020
17 Shuai B, Berneshawi A, Li X, et al. Siammot: siamese multi-object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021. 12372–12382
18 Gu A, Goel K, Ré C. Efficiently modeling long sequences with structured state spaces. ArXiv:2111.00396
19 Smith J T, Warrington A, Linderman S. Simplified state space layers for sequence modeling. In: Proceedings of the 11th International Conference on Learning Representations, 2023
20 Gu A, Dao T. Mamba: linear-time sequence modeling with selective state spaces. ArXiv:2312.00752
21 Ge Z, Liu S, Wang F, et al. Yolox: exceeding yolo series in 2021. ArXiv:2107.08430

22 Milan A, Leal-Taixé L, Reid I, et al. Mot16: a benchmark for multi-object tracking. ArXiv:1603.00831
23 Luiten J, Osep A, Dendorfer P, et al. HOTA: a higher order metric for evaluating multi-object tracking. Int J Comput Vis, 2021, 129: 548–578
24 Bewley A, Ge Z, Ott L, et al. Simple online and realtime tracking. In: Proceedings of IEEE International Conference on Image Processing (ICIP), 2016. 3464–3468
25 Wojke N, Bewley A, Paulus D. Simple online and realtime tracking with a deep association metric. In: Proceedings of IEEE International Conference on Image Processing (ICIP), 2017. 3645–3649
26 Long C, Haizhou A, Zijie Z, et al. Real-time multiple people tracking with deeply learned candidate selection and person re-identification. In: Proceedings of ICME, 2018
27 Aharon N, Orfaig R, Bobrovsky B Z. Bot-sort: robust associations multi-pedestrian tracking. ArXiv:2206.14651
28 Du Y, Song Y, Yang B, et al. Strongsort: make deepsort great again. ArXiv:2202.13514
29 Liu Z, Wang X, Wang C, et al. SparseTrack: multi-object tracking by performing scene decomposition based on pseudo-depth. IEEE Trans Circ Syst Video Technol, 2025, 35: 4870–4882
30 Zhang Y, Wang C, Wang X, et al. Robust multi-object tracking by marginal inference. In: Proceedings of European Conference on Computer Vision, 2022. 22–40
31 Yang M, Han G, Yan B, et al. Hybrid-sort: weak cues matter for online multi-object tracking. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2024. 6504–6512
32 Hu Y, Hua J, Han Z, et al. DiffusionMOT: a diffusion-based multiple object tracker. IEEE Trans Neural Netw Learn Syst, 2025, 36: 18203–18217
33 Shim K, Ko K, Yang Y, et al. Focusing on tracks for online multi-object tracking. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2025. 11687–11696
34 He L, Liao X, Liu W, et al. Fastreid: a PyTorch toolbox for general instance re-identification. ArXiv:2006.02631
35 Zhou K, Xiang T. Torchreid: a library for deep learning person re-identification in PyTorch. ArXiv:1910.10093
36 Zhou K, Yang Y, Cavallaro A, et al. Omni-scale feature learning for person re-identification. In: Proceedings of ICCV, 2019
37 Zhou K, Yang Y, Cavallaro A, et al. Learning generalisable omni-scale representations for person re-identification. IEEE Trans Pattern Anal Mach Intell, 2022, 44: 5056–5069
38 Leal-Taixé L, Milan A, Reid I, et al. MOTChallenge 2015: towards a benchmark for multi-target tracking. ArXiv:1504.01942
39 Dendorfer P, Rezatofighi H, Milan A, et al. Mot20: a benchmark for multi object tracking in crowded scenes. ArXiv:2003.09003
40 Du Y, Wan J, Zhao Y, et al. Giaotracker: a comprehensive framework for mcmot with global information and optimizing strategies in visdrone 2021. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops, 2021. 2809–2819
41 Yi K, Luo K, Luo X, et al. Ucmctrack: multi-object tracking with uniform camera motion compensation. In: Proceedings of the AAAI Conference on Artificial Intelligence, 2024. 6702–6710
42 Bergmann P, Meinhardt T, Leal-Taixé L. Tracking without bells and whistles. In: Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2019
43 Girshick R, Donahue J, Darrell T, et al. Rich feature hierarchies for accurate object detection and semantic segmentation. In: Proceedings of Computer Vision and Pattern Recognition (CVPR), 2014
44 Xiao C, Cao Q, Zhong Y, et al. MotionTrack: learning motion predictor for multiple object tracking. Neural Netws, 2024, 179: 106539
45 Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. In: Proceedings of Advances in Neural Information Processing Systems, 2017. 5998–6008
46 Dosovitskiy A, Beyer L, Kolesnikov A, et al. An image is worth 16x16 words: transformers for image recognition at scale. In: Proceedings of International Conference on Learning Representations, 2021
47 Lv W, Huang Y, Zhang N, et al. Diffmot: a real-time diffusion-based multiple object tracker with non-linear prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2024. 19321–19330
48 Dhariwal P, Nichol A. Diffusion models beat gans on image synthesis. In: Proceedings of Advances in Neural Information Processing Systems, 2021. 8780–8794
49 Xiao C, Cao Q, Luo Z, et al. Mambatrack: a simple baseline for multiple object tracking with state space model. In: Proceedings of the 32nd ACM International Conference on Multimedia, 2024
50 Zhu L, Liao B, Zhang Q, et al. Vision mamba: efficient visual representation learning with bidirectional state space model. In: Proceedings of the 41st International Conference on Machine Learning, 2024
51 Girshick R. Fast R-CNN. In: Proceedings of the IEEE International Conference on Computer Vision, 2015. 1440–1448
52 Rezatofighi H, Tsoi N, Gwak J, et al. Generalized intersection over union: a metric and a loss for bounding box regression. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019. 658–666
53 Bernardin K, Stiefelhagen R. Evaluating multiple object tracking performance: the clear mot metrics. EURASIP J Image Video Process, 2008, 2008: 1–10
54 Ristani E, Solera F, Zou R, et al. Performance measures and a data set for multi-target, multi-camera tracking. In: Proceedings of ECCV, 2016. 17–35
55 Cao J, Weng X, Khirodkar R, et al. Observation-centric sort: rethinking sort for robust multi-object tracking. ArXiv:2203.14360
56 Wu J, Cao J, Song L, et al. Track to detect and segment: an online multi-object tracker. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021. 12352–12361
57 Pang J, Qiu L, Li X, et al. Quasi-dense similarity learning for multiple object tracking. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021
58 Zhou X, Yin T, Koltun V, et al. Global tracking transformers. In: Proceedings of CVPR, 2022
59 Hu M, Zhu X, Wang H, et al. STDFormer: spatial-temporal motion transformer for multiple object tracking. IEEE Trans Circ Syst Video Technol, 2023, 33: 6571–6594
60 Huang H W, Yang C Y, Chai W, et al. Exploring learning-based motion models in multi-object tracking. ArXiv:2403.10826
61 Han X, Oishi N, Tian Y, et al. ETTrack: enhanced temporal motion predictor for multi-object tracking. Appl Intell, 2025, 55: 33
62 Chu P, Wang J, You Q, et al. Transmot: spatial-temporal graph transformer for multiple object tracking. ArXiv:2104.00194