MambaFlow: A Novel and Flow-Guided State Space Model for Scene Flow Estimation

Jiehao Luo, *Student Member, IEEE*, Jintao Cheng, Qingwen Zhang, *Graduate Student Member, IEEE*, Bohuan Xue, *Member, IEEE*, Rui Fan, *Senior Member, IEEE*, and Xiaoyu Tang, *Member, IEEE*

***Abstract*—Scene flow estimation aims to predict 3D motion from consecutive point cloud frames, which is of great interest in autonomous driving field. Existing methods face challenges such as insufficient spatio-temporal modeling and inherent loss of fine-grained feature during voxelization. However, the success of Mamba, a representative state space model (SSM) that enables global modeling with linear complexity, provides a promising solution. In this paper, we propose MambaFlow, a novel scene flow estimation network with a mamba-based decoder. It enables deep interaction and coupling of spatio-temporal features using a well-designed backbone. Innovatively, we steer the global attention modeling of voxel-based features with point offset information using an efficient Mamba-based decoder, learning voxel-to-point patterns that are used to devoxelize shared voxel representations into point-wise features. To further enhance the model's generalization capabilities across diverse scenarios, we propose a novel scene-adaptive loss function that automatically adapts to different motion patterns. Extensive experiments on the Argoverse 2 benchmark demonstrate that MambaFlow achieves state-of-the-art performance with real-time inference speed among existing works, enabling accurate flow estimation in real-world urban scenarios.**

***Index Terms*—Scene flow estimation, state space model (SSM), spatio-temporal deep coupling, real-time inference.**

Received 11 December 2024; revised 27 November 2025; accepted 2 February 2026. Date of publication 10 February 2026; date of current version 30 March 2026. This work was supported in part by the National Natural Science Foundation of China under Grant 62473288 and Grant 62233013, in part by the Guangdong Basic and Applied Basic Research Foundation under Grant 2024A1515012126, in part by the Fundamental Research Funds for the Central Universities, and in part by the Xiaomi Young Talents Program. *(Jiehao Luo and Jintao Cheng contributed equally to this work.) (Corresponding authors: Rui Fan; Xiaoyu Tang.)*

Jiehao Luo, Bohuan Xue, and Xiaoyu Tang are with the School of Data Science and Engineering, School of Electronics and Information Engineering, and Xingzhi College, South China Normal University, Shanwei 516600, China (e-mail: 20228132034@m.scnu.edu.cn; bxue@ieee.org; tangxy@scnu.edu.cn).

Jintao Cheng is with the Department of Electronic and Computer Engineering, Hong Kong University of Science and Technology, Hong Kong SAR, China (e-mail: jchengau@connect.ust.hk).

Qingwen Zhang is with the Division of Robotics, Perception, and Learning (RPL), KTH Royal Institute of Technology, SE-114 28 Stockholm, Sweden (e-mail: qingwen@kth.se).

Rui Fan is with the College of Electronic and Information Engineering, Shanghai Institute of Intelligent Science and Technology, Shanghai Research Institute for Intelligent Autonomous Systems, Shanghai Key Laboratory of Intelligent Autonomous Systems, State Key Laboratory of Autonomous Intelligent Unmanned Systems, and Frontiers Science Center for Intelligent Autonomous Systems (Ministry of Education), Tongji University, Shanghai 201804, China (e-mail: rui.fan@ieee.org).

The code is available at https://github.com/SCNU-RISLAB/MambaFlow.

Digital Object Identifier 10.1109/TIV.2026.3663171

I. Introduction

SCENE flow is a vector field that describes the motion of 3D points between two frames, representing the 3D counterpart of optical flow [1] in 2D images. The scene flow estimation task takes consecutive point cloud frames from sensors as input and outputs motion vectors for each point, providing autonomous perception systems with essential motion information for accurate environmental perception and decision-making [2], [3], [4], [5]. Moreover, scene flow estimation serves as a fundamental component for various downstream perception tasks, including object-level tasks such as 3D object detection [6] and point-level tasks such as moving object segmentation (MOS) [7], [8]. By providing dense point-wise motion vectors, scene flow offers rich motion information and geometric consistency constraints that are particularly beneficial for MOS and multi-object tracking tasks. This enables direct identification of dynamic points, offering geometry-based alternatives to purely semantic or appearance-based approaches, which is especially valuable for handling sparse LiDAR data in autonomous driving scenarios.

Existing scene flow estimation methods have made progress but still face several challenges. Current approaches [9], [10], [11], [12], [13], [14], [15] either directly concatenate consecutive point cloud frames for implicit temporal features extraction, leading to insufficient utilization of temporal information, or pursue efficiency by projecting point clouds into 2D pseudo-images, resulting in spatial feature loss. Jund et al. [13] proposed an end-to-end learning framework that concatenates consecutive point cloud frames and employs a flow embedding layer for feature extraction. Zhang et al. [14] drew inspiration from optical flow approaches and introduced a Gated Recurrent Unit (GRU) decoder for point cloud feature refinement. While these methods partially alleviate the feature loss, their reliance on only two consecutive frames for temporal information consistently limits model performance.

Temporal information plays a dominant role in scene flow estimation. This is effectively demonstrated by recent work Flow4D [16], which leverages more than two consecutive frames to achieve significantly improved performance on the Argoverse 2 benchmark. To reduce computational complexity, Flow4D decomposes 4D convolution into parallel temporal and spatial feature flow with 1D and 3D convolutions respectively. While this decomposition achieves computational efficiency, it sacrifices the holistic modeling of spatio-temporal correlations. The

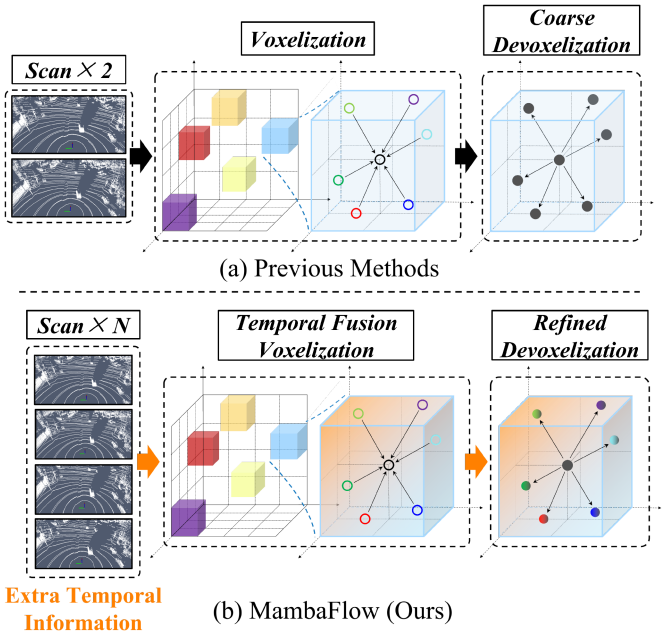

Fig. 1. Comparison of voxelization and devoxelization for scene flow estimation. (a) Previous methods typically use two consecutive frames with coarse devoxelization that assigns identical features to points in the same voxel, causing inherent feature loss. (b) Our MambaFlow leverages N consecutive scans for richer temporal information, with refined devoxelization that learns distinct voxel-to-point patterns for fine-grained feature representation.

assumption that temporal and spatial features can be processed independently ignores their inherent coupling in motion dynamics. Moreover, since temporal and spatial information characterizes fundamentally different aspects of motion dynamics and geometric structures, such simplified processing prevents effective feature interaction and temporal guidance.

To address the above challenges, we propose a novel multi-branch architecture for feature extraction that achieves deep coupling of spatio-temporal features. Specifically, our method incorporates a learnable fusion mechanism that dynamically weights temporal and spatial features while prioritizing temporal cues. This enables effective modeling of complex motion patterns across diverse scenarios, significantly enhancing the learning of motion dynamics from point clouds compared to existing methods.

Although our spatio-temporal coupling approach enables comprehensive feature extraction, the adopted voxel-based encoding introduces inherent feature loss, as points within the same voxel grid become indistinguishable, as shown in Fig. 1. Recent works [17], [18] demonstrate that transformer architectures can effectively learn point-wise correlations through global attention modeling. However, their quadratic inference complexity challenges real-time. Studies on SSM [19], [20], [21] suggest a promising alternative with linear-complexity global attention modeling. Drawing inspiration from recent advances in SSM for point cloud processing [7], [22], [23], we propose a Mamba-based decoder that steers global attention modeling of voxel-based features through point offset information. It achieves refined devoxelization by adaptively learning voxel-to-point patterns, which enables effective transformation from shared voxel representations to point-wise features, representing our core contribution to scene flow estimation.

As observed in [13], [14], scene flow estimation methods often exhibit limited generalization in autonomous driving scenarios due to the severe imbalance between dynamic and static point distributions, where over 90% of the point cloud exhibits zero displacement. Drawing insights from self-supervised approaches [13], [14], we propose a scene-adaptive loss function that leverages motion statistics derived from point displacement distributions.

Through extensive experiments, we demonstrate that our proposed feature extraction architecture integrated with the Mamba-based decoder, guided by scene-adaptive loss supervision, significantly enhances the model's dynamics-awareness and scene adaptability, achieving state-of-the-art performance on the Argoverse 2 benchmark among published works while maintaining real-time inference speed of 17.30 FPS. The main contributions of this paper are:

1) We propose MambaFlow, a novel SSM-based architecture for scene flow estimation that addresses voxelization feature loss through voxel-to-point pattern learning. To our knowledge, this represents the first application of state space modeling to scene flow estimation.
2) We propose a multi-branch backbone for deep coupling of spatio-temporal features, with an adaptive fusion strategy that prioritizes temporal information for complex motion patterns.
3) We propose a scene-adaptive loss function that leverages point displacement distributions to automatically distinguish between static and dynamic points without empirical thresholds.

We provide our code at https://github.com/SCNU-RISLAB/MambaFlow.

II. Related Work

A. Scene Flow Estimation Methods

Scene flow estimation in autonomous driving, while sharing similarities with object registration methods like DiffFlow3D [24] and [25] that achieve millimeter-level precision on small-scale datasets like ShapeNet [26] and FlyingThings3D [27], faces unique challenges when processing scenes from datasets like Argoverse 2 [28] and Waymo [29] containing 80 k-177 k points per frame, where substantial downsampling compromises practicality [13]. For such large-scale data, voxel-based encoding with efficient feature refinement emerges as the preferred approach to balance efficiency and feature preservation.

Many existing methods adopt flow embedding-based approaches [11], [13], [14], [30], establishing soft correspondences by concatenating spatial features from consecutive frames. However, this indirect feature correlation with two-frame input has been increasingly recognized as suboptimal for capturing temporal dynamics. Following successful practices in 3D detection [31], scene flow estimation has evolved towards multi-frame approaches. For example, Flow4D [16] processes five frames for

more robust estimation. Most recently, EulerFlow [32] reformulates scene flow as a continuous space-time PDE. However, to effectively exploit temporal dynamics while avoiding the computational complexity of continuous streams, we follow Flow4D's framework by adopting five consecutive frames as input with voxel-based encoding.

Recently, self-supervised methods have gained popularity due to the difficulty of obtaining scene flow ground truth. SeFlow [12] addresses key challenges through efficient dynamic classification and internal cluster consistency, while ICP-Flow [11] incorporates rigid-motion assumptions via histogram-based initialization and ICP alignment. However, precise motion estimation remains critical for autonomous driving safety, and state-of-the-art self-supervised methods like EulerFlow still show limitations in static background estimation, which could be catastrophic for autonomous driving systems. Although reducing annotation costs is appealing, the investment in labeled datasets is justified and necessary given the safety-critical nature of autonomous driving. We maintain a supervised architecture while incorporating insights from self-supervised approaches and propose a scene-adaptive loss function, which automatically extracts motion statistics from velocity histograms for better generalization without empirical thresholds.

B. State Space Model

SSMs [19], [20], [21] have gained attention as an efficient alternative to attention mechanisms and Transformer architectures, particularly for capturing long-term dependencies through hidden states. The Structured State Space Sequence (S4) [19] efficiently models long-range dependencies through diagonal parameterization, addressing computational bottlenecks in attention-based approaches. Building upon S4, researchers developed enhanced architectures such as S5 [20] and H3 [21]. Notably, Mamba [33] represents a significant breakthrough by introducing a data-dependent selective scan mechanism and integrating hardware-aware parallel scanning algorithms, establishing a novel paradigm distinct from CNNs and Transformers while maintaining linear computational complexity.

The linear complexity capability of Mamba has inspired extensive exploration in both vision [34], [35] and point cloud processing [7], [22], [23] domains. In 3D point cloud domain, recent works have demonstrated significant advances in adapting Mamba for various tasks. Mamba3D [22] introduced local norm pooling for geometric features and a bidirectional SSM design for enhancing both local and global feature representation. PointMamba [23] is one of the pioneers in applying state space models to point cloud analysis by utilizing space-filling curves for point tokenization with a non-hierarchical Mamba encoder, establishing a simple yet effective baseline for 3D vision applications. MambaMos [7] further explored SSM's potential in point cloud sequence modeling by adapting Mamba's selective scan mechanism for motion understanding, demonstrating that SSM's strong contextual modeling capabilities are particularly effective for capturing temporal correlations in moving object segmentation. These successes in achieving linear complexity while maintaining robust feature learning suggest promising potential for achieving fine-grained devoxelization in scene flow estimation.

Building upon these insights, we integrate the Mamba architecture into the scene flow estimation network to maintain real-time performance while avoiding the quadratic complexity of transformer-based approaches.

III. Preliminaries

A. State Space Model Background

State Space Model (SSM) is an advanced sequential model that demonstrates superior performance in modeling long-range dependencies in sequential data through hidden states. In the continuous-time domain, an SSM can be represented as a set of linear ordinary differential equations (ODEs) as (1):

$$\begin{aligned} h'(t) &= \mathbf{A}h(t) + \mathbf{B}x(t) \\ y(t) &= \mathbf{C}h(t) + Dx(t) \end{aligned} \tag{1}$$

where $h(t)$ denotes the hidden state, $x(t)$ and $y(t)$ denote the input and output sequences respectively, $\mathbf{A}$ denotes the state transition matrix of the system, $\mathbf{B}$ is the input projection matrix, $\mathbf{C}$ is the output projection matrix, and D is the skip connection parameter.

Mamba [33] is a sequence modeling framework based on SSM that introduces a selective scan mechanism to model complex state spaces through time-varying characteristics. Through a time scale parameter Δ, it makes the state transition matrix $\mathbf{A}$ and input projection matrix $\mathbf{B}$ input-dependent for selective feature filtering. The continuous system is discretized via Zero-Order Hold, which can be expressed as (2):

$$\begin{aligned} \bar{\mathbf{A}} &= e^{\Delta \mathbf{A}} \\ \bar{\mathbf{B}} &= (e^{\Delta \mathbf{A}} - \mathbf{I})\mathbf{A}^{-1}\Delta \mathbf{B} \end{aligned} \tag{2}$$

After discretization, the linear ordinary differential equations representing an SSM can be rewritten as (3):

$$\begin{aligned} h_t &= \bar{\mathbf{A}}h_{t-1} + \bar{\mathbf{B}}x_t \\ y_t &= \mathbf{C}h_t \end{aligned} \tag{3}$$

where h_t denotes the hidden state, x_t and y_t denote the input and output sequences respectively, and $\mathbf{C}$ is output projection matrix.

B. Scene Flow Problem Definition

Consider two consecutive point cloud frames $\mathcal{P}_t$ and $\mathcal{P}_{t+1}$ acquired at time instants t and $t+1$, with vehicle ego-motion transformation matrix $\mathbf{T}_{t,t+1}$. The scene flow estimation task aims to predict the motion vector $\hat{\mathbf{M}}_{t,t+1}(p) = (\Delta x, \Delta y, \Delta z)^{\mathrm{T}}$ for each point p in $\mathcal{P}_t$. We employ the End Point Error (EPE) as the evaluation metric, as defined in (4):

$$\mathrm{EPE}(p) = \|\hat{\mathbf{M}}(p) - \mathbf{M}_{gt}(p)\|_2 \tag{4}$$

where $\hat{\mathbf{M}}(p)$ and $\mathbf{M}_{gt}(p)$ denote the predicted and ground truth scene flow, respectively.

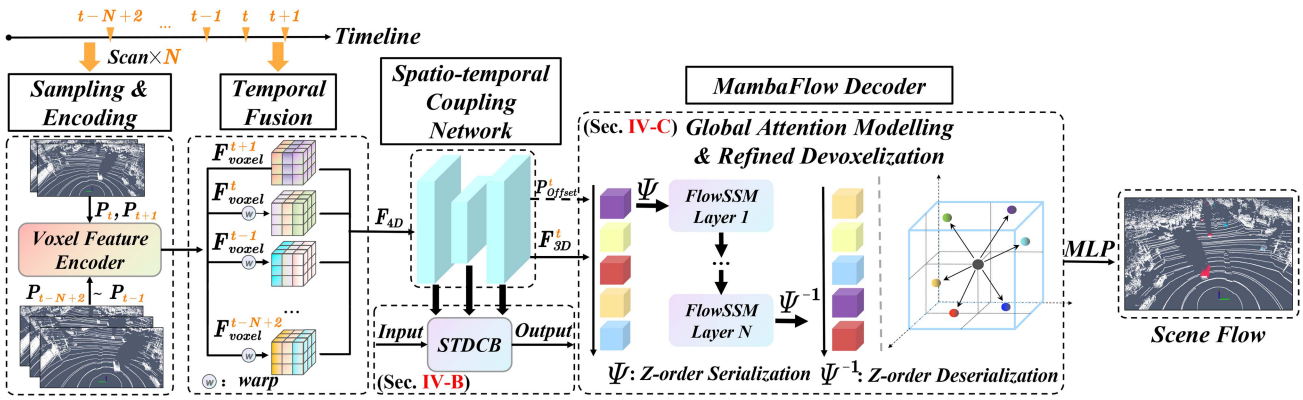

Fig. 2. Overall architecture of MambaFlow. The network first voxelizes and encodes five consecutive scans, forming 4D features by concatenating 3D voxel representations along the temporal dimension. These features are processed by our spatio-temporal coupling network for multi-scale feature learning. The decoder then learns voxel-to-point patterns through cascaded FlowSSM layers, enabling point-wise feature differentiation within the same voxel and generating the scene flow through an MLP layer.

The core objective of scene flow estimation is to minimize the average EPE, which can be expressed as (5):

$$\min \frac{1}{|\mathcal{P}_t|} \sum_{p \in \mathcal{P}_t} \|\hat{\mathbf{M}}(p) - \mathbf{M}_{gt}(p)\|_2 \tag{5}$$

where $|\mathcal{P}_t|$ denotes the number of points in $\mathcal{P}_t$. A method must address both dynamic objects in static environments and global scene changes induced by ego-motion. Such challenges necessitate effective integration of local features and global context for accurate 3D motion pattern capture.

IV. Methodology

A. MambaFlow Pipeline

1) Overall Architecture: As shown in Fig. 2, MambaFlow adopts an end-to-end architecture and generates scene flow estimation through three main stages: Spatio-temporal sampling and encoding, Spatio-temporal Deep Coupling Network, and MambaFlow Decoder.

In the Spatio-temporal Sampling and Encoding stage, N consecutive LiDAR scans are first transformed into 3D voxel representations through a voxel feature encoder, with all frames warped to the coordinate system of the time step $t+1$. The subsequent temporal fusion stage concatenates these 3D voxel representations along the temporal dimension, generating a 4D spatio-temporal feature tensor.

Next, the 4D tensor is processed by the Spatio-temporal Coupling Network. This network adopts a U-Net architecture with stacked Spatio-temporal Deep Coupling Block, which progressively learns multi-scale feature representations through deep hierarchical modeling. The network consists of a five-level encoder with stacking depths [2,2,2,2,2] and a four-level decoder with stacking depths [1,1,1,1], coupling multi-scale contextual information through sparse convolution operations. The fused features are residually combined with the input features to enhance feature representation.

For the devoxelization stage, the decoder is used to processes the extracted 3D voxel features from time step $t+1$. The voxel features are first serialized into sequences following space-filling curves. Through multiple cascaded FlowSSM modules, which incorporate discretized point offset information into state space for the global modeling of voxel-wise features, the decoder progressively refines voxel-wise features to point-wise features. Finally, the refined feature sequence is deserialized and fed into an MLP head to generate point-wise scene flow estimation.

2) Input Representation and Voxelization: To balance prediction performance and computational efficiency, we follow Flow4D [16] by sampling five consecutive point cloud frames as input.

Given consecutive point cloud frames $\mathcal{P}_\tau$ that can be represented as (6):

$$\begin{aligned} \mathcal{P}_\tau &= \{p_i \in \mathbb{R}^3\}_{i=0}^{N_\tau - 1}, \\ \tau &\in \{t-3, t-2, t-1, t, t+1\} \end{aligned} \tag{6}$$

where $p_i = (x_i, y_i, z_i)^{\mathrm{T}}$ denotes the point coordinates and N_τ denotes the number of points at time step τ.

Following previous studies, we first warp all point clouds to the perspective of time step $t+1$ using the known pose transformation matrices. For each transformed point cloud, we first employ an MLP to extract point-wise features $\boldsymbol{F}^{\tau}_{point}$, followed by a voxel feature encoder that aggregates these features into voxel-wise features $\boldsymbol{F}^{\tau}_{voxel}$. To form spatio-temporal representations, we extend $\boldsymbol{F}^{\tau}_{voxel}$ with a temporal dimension and concatenate them along the time axis, yielding a 4D voxel tensor $\boldsymbol{F}_{4D}$ that encodes both spatial and temporal information.

3) Serialization: Since our SSM-based decoder processes sequential data, we adopt Z-order space-filling curves as our serialization strategy inspired by [17]. We define a mapping function Ψ that projects 3D point cloud $\mathcal{P}_o$ to sequence $\mathcal{P}'_o$ while preserving spatial locality. The serialization and deserialization process can be expressed as (7):

$$\begin{aligned} \mathcal{P}'_o &= \Psi(\mathcal{P}_o) \\ \mathcal{P}_o &= \Psi^{-1}(\mathcal{P}'_o) \end{aligned} \tag{7}$$

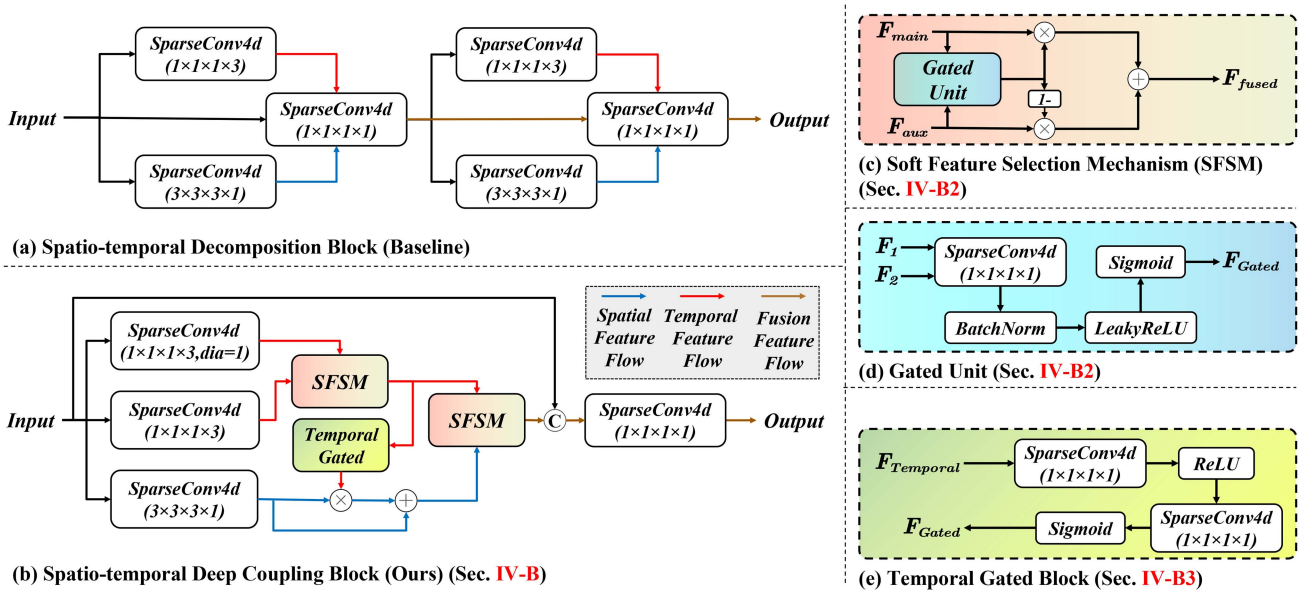

Fig. 3. Architecture of the Spatio-temporal Deep Coupling Block. (a) The baseline Spatio-temporal Decomposition Block [16] processes features through repeated convolutions at each stage. (b) Our proposed Spatio-temporal Deep Coupling Block achieves more efficient feature extraction by removing redundant convolutions modules and introducing a cross-timestep feature extraction. The right panel (c)-(e) shows the detailed structures of Soft Feature Selection Mechanism and gating mechanisms, illustrating the Spatial Feature Flow, Temporal Feature Flow, and Fused Feature Flow used in Spatio-temporal Deep Coupling Block.

B. Spatio-Temporal Deep Coupling Block

1) Feature Extraction Stage: As shown in Fig. 3, the proposed spatio-temporal deep coupling block (STDCB) consists of multiple sparse convolution branches, incorporating a cascaded Soft Feature Selection Mechanism to achieve adaptive feature interaction. Specifically, STDCB first sparsifies $\boldsymbol{F}_{4D}$ to obtain a sparse 4D tensor $\boldsymbol{F}_{\text{sparse}}$, which is then processed by three parallel branches. Drawing inspiration from the design of Flow4D [16], we adopt a convolution with kernel size $(3 \times 3 \times 3 \times 1)$ to extract geometric structures, and two convolutions with kernel size $(1 \times 1 \times 1 \times 3)$ to capture local motion patterns and cross-timestep dependencies through different receptive fields. The process of initial feature extraction can be formulated as (8):

$$\begin{aligned} \boldsymbol{F}_{spatial} &= \Phi_{3\times3\times3\times1}(\boldsymbol{F}_{sparse}) \\ \boldsymbol{F}_{temporal} &= \Phi_{1\times1\times1\times3}(\boldsymbol{F}_{sparse}) \\ \boldsymbol{F}^{ct}_{temporal} &= \Phi_{1\times1\times1\times3,dilation=1}(\boldsymbol{F}_{sparse}) \end{aligned} \tag{8}$$

where $\boldsymbol{F}_{spatial}$, $\boldsymbol{F}_{temporal}$ and $\boldsymbol{F}^{ct}_{temporal}$ denote the features extracted from spatial, local temporal, and cross-timestep temporal feature extraction paths respectively. Φ denotes the convolution kernel. For the cross-timestep, we employ a dilated convolution operation where a dilation factor of 1 is applied to expand the receptive field.

The decomposed design significantly reduces computational complexity compared to full 4D convolutions while preserving the ability to capture both spatial and temporal information.

2) Soft Feature Selection Mechanism: Given two feature tensors as input, Soft Feature Selection Mechanism designates one as the main feature flow $\boldsymbol{F}_{main}$ and the other as the auxiliary feature flow $\boldsymbol{F}_{aux}$ according to their roles in the specific task. Their relative importance is then computed through a Gated Unit. The attention weights $\boldsymbol{\alpha}$ can be formulated as (9):

$$\boldsymbol{\alpha} = \sigma(\text{LR}(\text{BN}(\Phi_p(\text{Concat}(\boldsymbol{F}_{main}, \boldsymbol{F}_{aux}))))) \tag{9}$$

where σ denotes the sigmoid activation, LR denotes the LeakyReLU, BN denotes the batch normalization operation, Φ_p denotes the point-wise convolution, and Concat denotes feature concatenation along the channel dimension.

The attention weights are then used to adaptively combine features from the two feature flow, where $\boldsymbol{\alpha}$ is multiplied with the main feature flow and $(1 - \boldsymbol{\alpha})$ with the auxiliary feature flow. This weighting scheme allows flexible control of feature importance: when $\boldsymbol{\alpha}$ is large, the main feature flow is emphasized, while smaller $\boldsymbol{\alpha}$ values give more weight to the complementary auxiliary feature flow.

3) Temporal Gated Block: After obtaining features from three parallel feature extraction paths, we first fuse temporal features by employing Soft Feature Selection Mechanism (SFSM) with consecutive temporal features as the main feature flow and cross-timestep temporal features as the auxiliary feature flow to complement cross-step temporal information, which can be formulated as (10):

$$\boldsymbol{F}'_{temporal} = \text{SFSM}(\boldsymbol{F}_{temporal}, \boldsymbol{F}^{ct}_{temporal}) \tag{10}$$

The fused temporal features then guide spatial feature learning through the Temporal Gated Block, where the attention weights $\boldsymbol{\beta}$ are computed as (11):

$$\boldsymbol{\beta} = \sigma(\Phi_p(\text{ReLU}(\Phi_p(\boldsymbol{F}'_{temporal})))) \tag{11}$$

The spatial features are modulated by temporal attention through gating and residual connection, which can be formulated

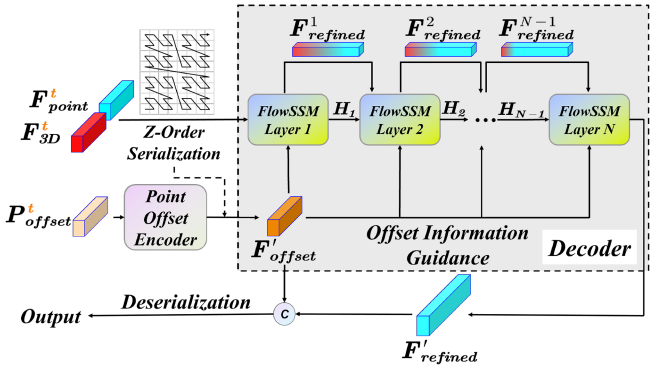

Fig. 4. MambaFlow Decoder architecture. Point-wise features and voxel features are first serialized through Z-order space-filling curves for spatial proximity preservation. The FlowSSM module consists of N cascaded FlowSSM layers, where point offset features guide the learning of voxel-to-point patterns in each layer for refined feature reconstruction. The final output is obtained through deserialization and feature fusion with point offset information.

as (12):

$$\boldsymbol{F}'_{spatial} = \boldsymbol{F}_{spatial} \odot (1 + \boldsymbol{\beta}) \tag{12}$$

where $\odot$ denotes element-wise multiplication.

Finally, $\boldsymbol{F}'_{spatial}$ is adaptively combined with $\boldsymbol{F}'_{temporal}$ using SFSM, followed by a residual connection with $\boldsymbol{F}_{sparse}$ and a point-wise convolution to fuse the features. The final output can be formulated as (13):

$$\boldsymbol{F}'_{sparse} = \Phi_p(\text{Concat}(\text{SFSM}(\boldsymbol{F}'_{temporal}, \boldsymbol{F}'_{spatial}); \boldsymbol{F}_{sparse})) \tag{13}$$

C. MambaFlow Decoder

The overall structure of the MambaFlow Decoder is shown in Fig. 4. During the decoding stage, we first extract the 3D voxel features $\boldsymbol{F}^t_{3D}$ at time step t from $\boldsymbol{F}'_{sparse}$. For each point in a voxel grid, we assign the corresponding voxel features as their initial point-wise representations, denoted as $\boldsymbol{F}^t_{coarse}$, where all points in the same voxel share identical feature values.

Meanwhile, we preserve both point-wise features $\boldsymbol{F}^t_{point}$ and point offset information $\boldsymbol{P}^t_{offset}$ during encoding stage. Considering that $\boldsymbol{P}^t_{offset}$ only have 3 channels, which may cause information imbalance when directly used as decoder input, we extend their feature dimension to a matching dimension of $\boldsymbol{F}^t_{3D}$ through a point offset encoder, yielding point-wise offset features $\boldsymbol{F}^t_{offset}$. The concatenation of $\boldsymbol{F}^t_{3D}$ and $\boldsymbol{F}^t_{point}$ along the channel dimension forms the initial input features $\boldsymbol{F}^0_{coarse}$ to the first decoding layer, with $\boldsymbol{F}^t_{offset}$ serving as guiding information for feature refinement.

The key component of our decoder is FlowSSM, a variant of SSM. As shown in Algorithm 1, it incorporates discretized point offset information into state space for global modeling of voxel-wise features. To enable sequence modeling, we first serialize both coarse features and offset features using Z-order space-filling curves, which can be formulated as (14):

$$\begin{aligned} \boldsymbol{F}'_{coarse} &= \Psi(\boldsymbol{F}^t_{coarse}) \\ \boldsymbol{F}'_{offset} &= \Psi(\mathbf{P}^t_{offset}) \end{aligned} \tag{14}$$

Algorithm 1: FlowSSM Process.

Require: Coarse flow features $\boldsymbol{F}^{i-1}_{coarse}$: $(B, N, 2C)$
Point offset features $\boldsymbol{F}'_{offset}$: (B, N, C)
Hidden state $\boldsymbol{H}_{i-1}$: (B, D, N)
Ensure: Refined flow features $\boldsymbol{F}^i_{coarse}$: (B, N, C)
Updated hidden state $\boldsymbol{H}_i$: (B, D, N)
/* Parameterize data independent matrices */
$\mathbf{A}$: $(D, N) \leftarrow$ Parameter
D: (scalar) $\leftarrow$ Parameter
/* Parameterize data dependent matrices via point offset features */
Δ: (B, D), $\mathbf{B}$: (B, N), $\mathbf{C}$: $(B, N) \leftarrow \text{Linear}(\boldsymbol{F}'_{offset})$
/* Discretize */
$\bar{\mathbf{A}}$: $(B, D, N) \leftarrow \text{Exp}(\Delta \otimes \mathbf{A})$
$\bar{\mathbf{B}}$: $(B, D, N) \leftarrow \Delta \otimes \mathbf{B}$
/* Running SSM */
$\boldsymbol{F}^i_{coarse}$: $(B, N, 2C)$, $\boldsymbol{H}^i$: (B, D, N)
$\leftarrow \text{SSM}(\bar{\mathbf{A}}, \bar{\mathbf{B}}, \mathbf{C}, \mathbf{D})(\boldsymbol{F}^{i-1}_{coarse}, \boldsymbol{H}_{i-1})$
Return: $\boldsymbol{F}^i_{refind}, \boldsymbol{H}_i$

The decoder then progressively refines these features through multiple cascaded FlowSSM layers, where each layer conditions the state space matrices on $\boldsymbol{F}'_{offset}$ for adaptive feature refinement. After N iterations, the refined feature sequence is first deserialized to obtain refined point-wise features $\boldsymbol{F}'_{refined}$, then concatenated with $\boldsymbol{F}'_{offset}$. Finally, the scene flow estimation $\hat{\mathbf{M}}_{t,t+1}(p)$ is generated through deserialization and an MLP head.

D. Scene-Adaptive Loss

In scene flow estimation, supervised methods relying on manually labeled data often show limited generalization ability to real-world scenes. Inspired by the loss function design of FastFlow3D [13], DeFlow [14] divides points into three subsets using velocity thresholds of 0.4 m/s and 1.0 m/s, and computes the total loss by averaging the endpoint errors within each subset, which can be formulated as (15):

$$\mathcal{L} = \sum_{i=1}^{3} \frac{1}{|\mathcal{P}_i|} \sum_{p \in \mathcal{P}_i} \left\| \Delta\hat{\mathbf{M}}(p) - \Delta\mathbf{M}_{\text{gt}}(p) \right\|_2, \tag{15}$$

where $\Delta\hat{\mathbf{M}}(p)$ and $\Delta\mathbf{M}_{\text{gt}}(p)$ denote the predicted and ground-truth motion of point p, respectively.

However, using a single fixed threshold to separate static and dynamic points can misclassify slow-moving objects across diverse traffic scenarios. Considering that over 90% of points exhibit near-zero motion in autonomous driving, directly minimizing the average endpoint error leads to an objective dominated by quasi-static regions, causing the network to underfit safety-critical dynamic points. We propose a scene-adaptive loss that automatically determines the static/dynamic separation threshold based on the empirical motion distribution of each scene, while retaining a fixed upper threshold to distinguish motion scales.

Let r denote the point-wise motion magnitude and $F(r)$ be its cumulative distribution function. We define the tail probability as (16):

$$\mathcal{T}(r) = \mathbb{P}(r' > r) = 1 - F(r), \tag{16}$$

where r' denotes the motion magnitude of an arbitrary point. For a tail-mass level $\tau \in (0, 1)$, the scene-adaptive threshold is defined as (17):

$$r_\tau = \inf \{r \ : \ \mathcal{T}(r) \leq \tau\}, \tag{17}$$

where r_τ identifies the motion level above which only a fraction τ of points remain. This quantile-based threshold adapts to the scene-specific motion distribution, separating the dense near-static majority from the sparse dynamic tail.

To further distinguish motion scales among dynamic points, we introduce a characteristic threshold r_c that separates moderate and high-speed motions. Unlike the lower threshold r_τ which varies across scenes, r_c is set to a fixed value based on typical urban traffic speeds, as the distinction between moderate and high-speed motions is relatively consistent across different scenarios. This yields a three-way partition as shown in (18):

$$\begin{cases} \mathcal{P}_0 = \{p \in \mathcal{P} \mid r(p) \leq r_\tau\}, \\ \mathcal{P}_1 = \{p \in \mathcal{P} \mid r_\tau < r(p) \leq r_c\}, \\ \mathcal{P}_2 = \{p \in \mathcal{P} \mid r(p) > r_c\}, \end{cases} \tag{18}$$

where $\mathcal{P}_0$, $\mathcal{P}_1$, and $\mathcal{P}_2$ correspond to quasi-static background, moderate motions, and large motions, respectively. The scene-adaptive loss is then computed as (19)

$$\mathcal{L}_{\text{total}} = \sum_{i=0}^{2} \frac{1}{|\mathcal{P}_i|} \sum_{p \in \mathcal{P}_i} \left\| \Delta\hat{\mathbf{M}}(p) - \Delta\mathbf{M}_{\text{gt}}(p) \right\|_2, \tag{19}$$

where $|\mathcal{P}_i|$ denotes the number of points in the i-th regime. By computing the loss over the three motion regimes separately, this formulation resembles a cost-sensitive scheme where each regime contributes comparably despite highly imbalanced sample sizes, enabling the model to learn motion patterns across the full spectrum of scene dynamics.

V. Experiment

A. Experiment Setups

1) Datasets: The proposed method is evaluated on the Argoverse 2 dataset [28], a large-scale autonomous driving benchmark containing 1,000 diverse scenarios. Each sequence in this dataset spans 15 seconds and encompasses 30 distinct object classes in complex urban environments. The dataset provides comprehensive sensor data, including LiDAR scans, detailed 3D annotations, and HD maps, making it particularly suitable for evaluating scene flow estimation in real-world urban scenarios.

2) Metrics: We employ two metrics to evaluate our method. First, we use the standard End Point Error (EPE), which measures the L2 distance between predicted and ground truth scene flow vectors. However, as EPE tends to be dominated by large objects and fails to reflect performance on smaller, safety-critical objects like pedestrians, we also adopt Bucket Normalized EPE [15]. This metric evaluates each object category separately and normalizes the error by object speed, enabling more balanced assessment across different object types and motion patterns.

3) Implementation Details: Our model is implemented in PyTorch with a two-phase training strategy. The first phase focuses on training the backbone network with scene-adaptive loss without the decoder to learn robust feature representations. In the second phase, we integrate the decoder and fine-tune the entire architecture with a lower learning rate to preserve the pre-trained backbone knowledge while optimizing decoder performance.

We conduct distributed training on 8 NVIDIA RTX 4090 GPUs. The first phase runs for 30 epochs with an initial learning rate of 2e-4 and batch size of 5 per GPU. The second phase continues for 50 epochs with an initial learning rate of 2e-6 and batch size of 3 per GPU, while maintaining other hyperparameters.

B. Scene Flow Estimation Performance

1) Quantitation Analysis: We compare our proposed MambaFlow with various supervised and self-supervised scene flow methods. Table I presents the comparative results evaluated through the official Argoverse 2 test server. The proposed method achieves state-of-the-art performance in all EPE metrics, demonstrating the most accurate point-level motion estimation across all published works. Specifically, MambaFlow surpasses Flow4D by significant margins in all three metrics, showing a 14.7% improvement in average EPE, 8.9% in foreground dynamic estimation, and 68.1% in background static prediction.

For object-level evaluation using Bucketed Normalized EPE, our method achieves competitive performance in dynamic object estimation, particularly excelling in rigid objects such as cars and other vehicles. MambaFlow's balanced performance across both static and dynamic scenarios, especially its superior point-level accuracy as demonstrated by 3-way EPE metrics, validates its effectiveness as a comprehensive scene flow estimation solution.

As shown in Table II, we compare our method with several SoTA supervised methods on the Argoverse 2 validation set. MambaFlow achieves the best overall performance across all metrics. Notably, even with only 2 frames, MambaFlow maintains competitive performance compared to other 5-frame methods, demonstrating the effectiveness of our temporal modeling. The 5-frame configuration represents a balanced trade-off between accuracy and efficiency for practical deployment.

2) Qualitative Analysis: As shown in Fig. 5, MambaFlow demonstrates superior performance across diverse challenging scenarios. In the pedestrian crossing scene (Row 1), our method accurately captures fine-grained motion details while correctly identifying static pedestrians, whereas competing methods struggle with such subtle distinctions. For distant vehicle prediction (Row 2), MambaFlow maintains accurate flow estimation even for far-field objects where other methods show significant degradation. Most notably, in high-speed scenarios (Row 3), our method exhibits strong rigid-body consistency

TABLE I
QUANTITATIVE COMPARISON ON THE ARGOVERSE 2 TEST SET. 'SUP.' REPRESENTS SUPERVISED METHODS. 'FD' REPRESENTS FOREGROUND DYNAMIC, 'BS' REPRESENTS BACKGROUND STATIC, AND 'FS' REPRESENTS FOREGROUND STATIC.

Method	Sup.	3-way Endpoint Error (↓)				Bucketed Normalized Endpoint Error (↓)					
						Dynamic					Static mean
		Avg.	FD	BS	FS	mean	Car	Other vehicles	Pedes-trian	Wheeled-vru	
NSFP [9]		0.0606	0.1158	0.0344	0.0316	0.4219	0.2509	0.3313	0.7225	0.3831	0.0279
FastNSF [10]		0.1118	0.1844	0.0907	0.0814	0.3826	0.2901	0.4126	0.5002	0.3215	0.0736
ICP-Flow [11]		0.0650	0.1369	0.0250	0.0332	0.3309	0.1945	0.3314	0.4353	0.3626	0.0271
SeFlow [12]		0.0536	0.1323	0.0043	0.0242	0.3194	0.2178	0.3464	0.4452	0.2683	0.0148
EulerFlow [36]		0.0423	0.0498	0.0526	0.0245	**0.1303**	0.0929	0.1408	**0.1947**	**0.0931**	0.0253
FastFlow3D [13]	✓	0.0735	0.1917	0.0027	0.0262	0.5323	0.2429	0.3908	0.9818	0.5139	0.0182
DeFlow [14]	✓	0.0501	0.1091	0.0062	0.0352	0.3704	0.1530	0.3150	0.6615	0.3520	0.0262
TrackFlow [15]	✓	0.0473	0.1030	0.0024	0.0365	0.2689	0.1817	0.3054	0.3581	0.2302	0.0447
Flow4D [16]	✓	0.0224	0.0494	0.0047	0.0130	0.1454	0.0871	0.1505	0.2165	0.1272	0.0106
MambaFlow (Ours)	✓	**0.0191**	**0.0450**	**0.0015**	**0.0108**	0.1422	**0.0786**	**0.1399**	0.2369	0.1136	**0.0077**

TABLE II
QUANTITATIVE COMPARISON ON THE ARGOVERSE 2 VALIDATION SET. '#F' REPRESENTS THE NUMBER OF INPUT FRAMES USED BY EACH METHOD.

Method	#F	Bucketed Normalized EPE (↓)		3-way Endpoint Error (↓)				Dynamic IoU (↑)
		mean Dynamic	mean Static	Avg.	FD	BS	FS	
FastFlow3D [13]	2	0.3975	0.0113	0.0461	0.1218	0.0028	0.0136	0.7169
DeFlow [14]	2	0.3299	0.0192	0.0446	0.1013	0.0043	0.0281	0.7505
Flow4D [16]	5	0.1775	0.0092	0.0206	0.0468	0.0033	0.0117	0.8137
SSF [37]	2	0.2318	0.0157	0.0341	0.0843	0.0038	0.0143	0.7945
DeltaFlow [38]	5	0.1637	0.0090	0.0232	0.0555	0.0038	0.0104	0.8585
MambaFlow (Ours)	2	0.1688	0.0093	0.0207	0.0485	0.0035	0.0102	0.8572
MambaFlow (Ours)	5	**0.1620**	**0.0079**	**0.0176**	**0.0414**	**0.0018**	**0.0096**	**0.8661**

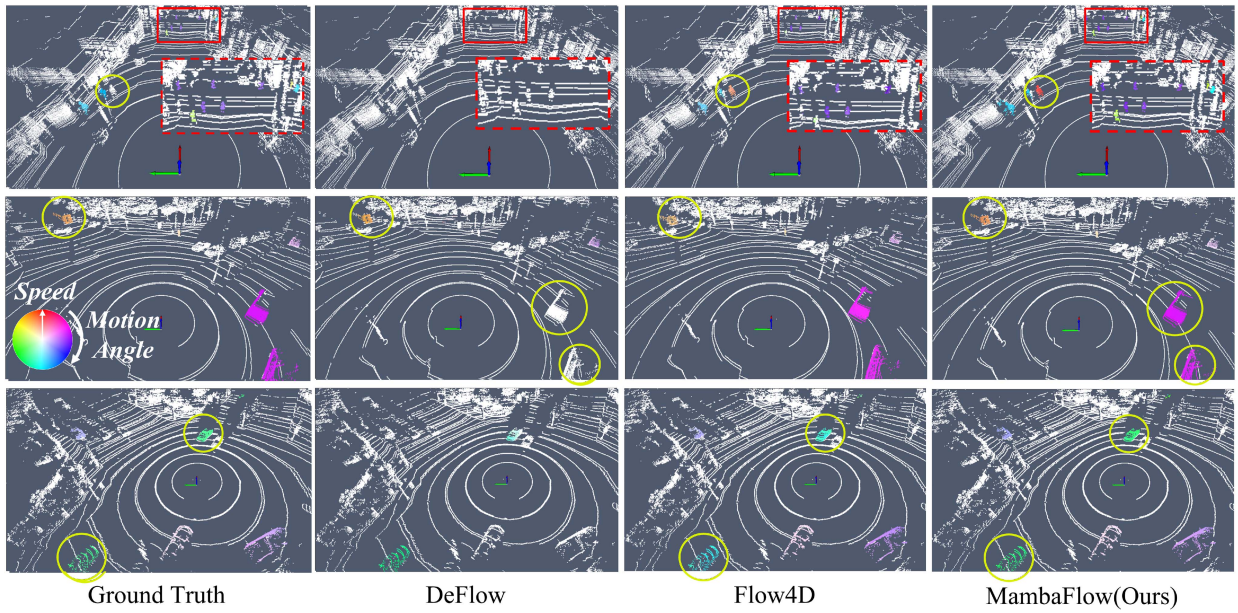

Fig. 5. Qualitative results on the Argoverse 2 validation set. From left to right: Ground Truth, DeFlow, Flow4D, and our proposed MambaFlow. The color legend indicates both speed (shown by color intensity) and motion angle (2D), aligned with the vehicle's forward direction. Row 1: Complex pedestrian crossing scenario where MambaFlow accurately distinguishes fine-grained pedestrian features and correctly identifies static walking pedestrians as annotated in ground truth. Row 2: Urban scene with distant vehicles where MambaFlow precisely predicts scene flow for far-field objects. Row 3: High-speed vehicle scenario demonstrating MambaFlow's superior rigid-body flow consistency across the entire vehicle structure. Yellow circles highlight regions where our method shows superior performance.

TABLE III
PERFORMANCE COMPARISON OF DIFFERENT COMPONENT CONFIGURATIONS. 'STDCB' REPRESENTS THE SPATIO-TEMPORAL DEEP COUPLING BLOCK, 'M.F. DECODER' REPRESENTS THE MAMBAFLOW DECODER AND 'S.A. LOSS' REPRESENTS THE SCENE-ADAPTIVE LOSS.

Method	Components			B.N. EPE (↓)	
	STDCB	M.F. Decoder	S.A. Loss	mean D.	mean S.
Baseline	-	-	-	0.1775	0.0092
(i)	✓			0.1685	0.0087
		✓		0.1720	0.0091
			✓	0.1744	0.0078
(ii)	✓	✓		0.1646	0.0090
	✓		✓	0.1645	0.0080
		✓	✓	0.1695	0.0085
MambaFlow	✓	✓	✓	**0.1620**	**0.0079**

TABLE IV
ABLATION STUDY ON FLOWSSM ITERATION COUNT. '#ITR' REPRESENTS THE DIFFERENT NUMBERS OF ITERATIONS.

#Itr	Bucketed Normalized EPE (↓)		Efficiency
	Mean Dynamic	Mean Static	FPS
1	**0.1620**	**0.0079**	**17.30**
2	0.1645	0.0079	16.66
3	0.1697	0.0081	15.99
4	0.1725	0.0096	14.90
5	0.1655	0.0092	13.76

across entire vehicle structures, producing coherent flow patterns that align well with ground truth, while baseline methods show fragmented or inconsistent predictions. These results validate MambaFlow's robustness in handling both fine-grained motion details and large-scale rigid motion patterns.

C. Ablation Study

The component analysis in Table III demonstrates the contribution of each proposed module. In method (i), for both mean Dynamic EPE and mean Static EPE metrics, incorporating STDCB alone brings improvements of 5.1% and 5.4% respectively, validating its effectiveness in temporal-spatial feature coupling. The application of MambaFlow decoder yields improvements of 3.1% and 1.1% respectively, highlighting its capability in feature recovery. Meanwhile, utilizing Scene-adaptive Loss alone for supervision training improves mean Dynamic EPE and mean Static EPE by 1.7% and 15.2% respectively, demonstrating its robust scene adaptation capability. In method (ii), when validating pairwise combinations of components, the model's performance shows substantial improvements, particularly with the synergistic effect of STDCB and scene-adaptive loss, which yields improvements of 7.3% and 13.0% in mean Dynamic EPE and mean Static EPE respectively. By combining all components, the proposed MambaFlow achieves the best performance.

TABLE V
GENERALIZATION OF SCENE-ADAPTIVE LOSS ACROSS DIFFERENT METHODS. WE EVALUATE THE EFFECTIVENESS OF OUR PROPOSED SCENE-ADAPTIVE LOSS FUNCTION BY APPLYING IT TO DIFFERENT BASELINE METHODS ON THE ARGOVERSE 2 VALIDATION SET.

Method	S.A. Loss	B.N. EPE (↓)	
		mean D.	mean S.
FastFlow3D [13]	-	0.3975	0.0113
FastFlow3D [13]	✓	0.3871	0.0105
Δ Improvement		0.0104 (2.6%)	0.0008 (7.1%)
DeFlow [14]	-	0.3299	0.0192
DeFlow [14]	✓	0.2623	0.0141
Δ Improvement		0.0676 (20.5%)	0.0051 (26.6%)
MambaFlow	-	0.1775	0.0092
MambaFlow	✓	0.1744	0.0078
Δ Improvement		0.0031 (1.7%)	0.0014 (15.1%)

As shown in Table IV, the results demonstrate that with a single iteration, our method achieves an optimal balance between accuracy and computational speed. The computational complexity of FlowSSM is $O(N \cdot D)$ where N is the sequence length and D is the hidden dimension, which is linear compared to Transformer-based attention mechanisms with $O(N^2 \cdot D)$ complexity, making it particularly efficient for processing large-scale point clouds. We also tested the effect of increasing the number of iterations, and the results show that the increasing number of iterations not only fails to improve performance but also leads to slight degradation in computational efficiency. Therefore, we adopt single iteration in our overall architecture. This suggests that a single well-designed SSM layer with sufficient capacity effectively captures motion patterns without requiring deep stacking, attributed to the linear nature of motion in short time intervals. However, deeper SSM architectures may prove beneficial for more complex tasks such as long-term trajectory prediction or multi-object tracking.

To validate the generalization capability of our scene-adaptive loss function, we apply it to different baseline methods as shown in Table V. The results demonstrate that our loss function consistently enhances dynamic motion perception across diverse architectures: FastFlow3D [13] achieves 2.6% improvement in mean dynamic EPE, DeFlow [14] achieves 20.5% improvement, and MambaFlow achieves 1.7% improvement. These consistent gains across methods with different architectural designs validate the strong generalization capability of our scene-adaptive loss, demonstrating its effectiveness in enhancing dynamic object perception.

D. Evaluation of Consumption

As shown in Table VI, our method achieves superior performance with only 3.1 M parameters, representing a 32.6% reduction in parameter count compared to Flow4D. Instead of using STDB with redundant convolution operations as in Flow4D, STDCB eliminates duplicate feature extraction processes based on our observation and experimental validation that features extracted in earlier stages already contain sufficient information for

TABLE VI
RESOURCE CONSUMPTION OF DIFFERENT METHODS

Methods	FPS	Params(M)	Total Size(MB)	GM(GiB)
FastFlow3D	**20.35**	6.8	27.262	2.37
DeFlow	19.82	6.9	27.568	2.42
Flow4D	14.68	4.6	18.412	**1.78**
MambaFlow	17.30	**3.1**	**12.598**	2.04

subsequent fusion. This architectural optimization also improves computational efficiency, enabling our method to achieve 17.30 FPS compared to Flow4D's 14.68 FPS.

In terms of memory usage, our method achieves a better balance between performance and resource efficiency with moderate memory usage of 2.04 GiB. This slight increase in memory overhead is well compensated by the substantial performance improvements. With a more compact model size of 12.598 MB compared to Flow4D's 18.412 MB, MambaFlow is particularly suitable for deployment in resource-constrained scenarios while maintaining state-of-the-art performance.

VI. CONCLUSION

In this paper, we present MambaFlow, a novel SSM-based scene flow estimation approach that introduces a Mamba decoder for refined devoxelization through voxel-to-point pattern learning. Extensive experiments demonstrate state-of-the-art performance on the Argoverse 2 benchmark while maintaining real-time inference capability.

Despite these achievements, our method assumes accurate pose transformations between consecutive LiDAR frames, which may limit applicability in scenarios with uncertain pose estimation. Future work could explore integrating robust pose estimation into the pipeline for enhanced robustness in challenging scenarios.

REFERENCES

[1] Z. Yi et al., "FocusFlow: Boosting key-points optical flow estimation for autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 9, no. 1, pp. 2794–2807, Jan. 2024.

[2] H. Liu, Z. Huang, X. Mo, and C. Lv, "Augmenting reinforcement learning with transformer-based scene representation learning for decision-making of autonomous driving," *IEEE Trans. Intell. Veh.*, vol. 9, no. 3, pp. 4405–4421, Mar. 2024.

[3] H. Shi, Q. Jiang, K. Yang, X. Yin, Z. Wang, and K. Wang, "Beyond the field-of-view: Enhancing scene visibility and perception with clip-recurrent transformer," *IEEE Trans. Intell. Veh.*, vol. 10, no. 2, pp. 775–793, Feb. 2025.

[4] J. Luo et al., "OverlapMamba: A shift state space model for LiDAR-based place recognition," *IEEE Robot. Automat. Lett.*, vol. 10, no. 8, pp. 8380–8387, Aug. 2025.

[5] J. Cheng et al., "Scale, don't fine-tune: Guiding multimodal LLMs for efficient visual place recognition at test-time," *IFAC-PapersOnLine*, vol. 59, no. 35, pp. 97–102, 2025.

[6] H. Meng, C. Li, G. Chen, L. Chen, and A. Knoll, "Efficient 3D object detection based on pseudo-LiDAR representation," *IEEE Trans. Intell. Veh.*, vol. 9, no. 1, pp. 1953–1964, Jan. 2024.

[7] K. Zeng et al., "MambaMOS: LiDAR-based 3 D moving object segmentation with motion-aware state space model," in *Proc. 32nd ACM Int. Conf. Multimedia*, 2024, pp. 1505–1513.

[8] J. Cheng et al., "MF-MOS: A motion-focused model for moving object segmentation," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 12499–12505.

[9] X. Li, J. Kaesemodel Pontes, and S. Lucey, "Neural scene flow prior," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, vol. 34, pp. 7838–7851.

[10] X. Li, J. Zheng, F. Ferroni, J. K. Pontes, and S. Lucey, "Fast neural scene flow," in *Proc. IEEE/CVF Int. Conf. Comput. Vis.*, 2023, pp. 9878–9890.

[11] Y. Lin and H. Caesar, "ICP-Flow: LiDAR scene flow estimation with ICP," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 15501–15511.

[12] Q. Zhang, Y. Yang, P. Li, O. Andersson, and P. Jensfelt, "SeFlow: A self-supervised scene flow method in autonomous driving," in *Proc. Eur. Conf. Comput. Vis.*, 2024, pp. 353–369.

[13] P. Jund, C. Sweeney, N. Abdo, Z. Chen, and J. Shlens, "Scalable scene flow from point clouds in the real world," *IEEE Robot. Automat. Lett.*, vol. 7, no. 2, pp. 1589–1596, Apr. 2022, doi: 10.1109/lra.2021.3139542.

[14] Q. Zhang, Y. Yang, H. Fang, R. Geng, and P. Jensfelt, "DeFlow: Decoder of scene flow network in autonomous driving," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2024, pp. 2105–2111.

[15] I. Khatri, K. Vedder, N. Peri, D. Ramanan, and J. Hays, "I can't believe it's not scene flow!," in *Proc. Eur. Conf. Comput. Vis.*, 2025, pp. 242–257.

[16] J. Kim, J. Woo, U. Shin, J. Oh, and S. Im, "Flow4D: Leveraging 4D voxel network for LiDAR scene flow estimation," *IEEE Robot. Automat. Lett.*, vol. 10, no. 4, pp. 3462–3469, Apr. 2025.

[17] X. Wu et al., "Point transformer V3: Simpler faster stronger," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 4840–4851.

[18] J. Li, Y. Zhang, P. Yun, G. Zhou, Q. Chen, and R. Fan, "RoadFormer: Duplex transformer for RGB-normal semantic road scene parsing," *IEEE Trans. Intell. Veh.*, vol. 9, no. 7, pp. 5163–5172, Jul. 2024.

[19] A. Gu, K. Goel, and C. Ré, "Efficiently modeling long sequences with structured state spaces," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2022.

[20] J. T. Smith, A. Warrington, and S. W. Linderman, "Simplified state space layers for sequence modeling," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2023.

[21] D. Y. Fu et al., "Hungry hungry hippos: Towards language modeling with state space models," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2023.

[22] X. Han, Y. Tang, Z. Wang, and X. Li, "Mamba3D: Enhancing local features for 3D point cloud analysis via state space model," in *Proc. ACM Int. Conf. Multimedia (ACM MM)*, 2024, pp. 4995–5004.

[23] D. Liang et al., "PointMamba: A simple state space model for point cloud analysis," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2024, vol. 37, pp. 32653–32677.

[24] J. Liu et al., "Difflow3D: Toward robust uncertainty-aware scene flow estimation with iterative diffusion-based refinement," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2024, pp. 15109–15119.

[25] Z. Wang, Y. Wei, Y. Rao, J. Zhou, and J. Lu, "3D point-voxel correlation fields for scene flow estimation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 45, no. 11, pp. 13621–13635, Nov. 2023.

[26] A. Chang et al., "ShapeNet: An information-rich 3D model repository," 2015, *arXiv:1512.03012*.

[27] N. Mayer et al., "A large dataset to train convolutional networks for disparity, optical flow, and scene flow estimation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2016, pp. 4040–4048.

[28] B. Wilson et al., "Argoverse 2: Next generation datasets for self-driving perception and forecasting," in *Proc. Neural Inf. Process. Syst. Track Datasets Benchmarks*, J. Vanschoren and S. Yeung, Eds., 2021, vol. 1, pp. 1–27. [Online]. Available: https://datasets-benchmarks-proceedings.neurips.cc/paper_files/paper/2021/file/4734ba6f3de83d861c3176a6273cac6d-Paper-round2.pdf

[29] P. Sun et al., "Scalability in perception for autonomous driving: Waymo open dataset," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 2446–2454.

[30] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning scene flow in 3D point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2019, pp. 529–537.

[31] K. Vedder and E. Eaton, "Sparse pointpillars: Maintaining and exploiting input sparsity to improve runtime on embedded systems," in *Proc. 2022 IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2022, pp. 2025–2031.

[32] K. Vedder et al., "Scene flow as a partial differential equation," 2024, *arXiv:2410.02031*.

[33] A. Gu and T. Dao, "Mamba: Linear-time sequence modeling with selective state spaces," in *Proc. 1st Conf. Lang. Model. (COLM)*, 2024.

[34] L. Zhu et al., "Vision Mamba: Efficient visual representation learning with bidirectional state space model," in *Proc. Int. Conf. Mach. Learn. (ICML)*, 2024, pp. 62429–62442.

[35] X. Pei, T. Huang, and C. Xu, "EfficientvMamba: Atrous selective scan for light weight visual Mamba," in *Proc. AAAI Conf. Artif. Intell.*, 2025, vol. 39, no. 6, pp. 6443–6451.
[36] K. Vedder et al., "Neural eulerian scene flow fields," in *Proc. Int. Conf. Learn. Representations (ICLR)*, 2025.
[37] T. Song, Y. Liu, Z. Yao, and X. Wu, "SSF-MOS: Semantic scene flow assisted moving object segmentation for autonomous vehicles," *IEEE Trans. Instrum. Meas.*, vol. 73, 2024, Art. no. 2507112.
[38] Q. Zhang et al., "DeltaFlow: An efficient multi-frame scene flow estimation method," in *Proc. Adv. Neural Inf. Process. Syst. (NeurIPS)*, 2025.

Jiehao Luo (Student Member, IEEE) is currently working toward the B.S. degree in Internet of Things engineering with the School of Data Science and Engineering, and Xingzhi College, South China Normal University, Guangzhou, China. His research interests include computer vision and robotic perception. He is a Student Member of the Chinese Association for Artificial Intelligence.

Jintao Cheng received the bachelor's degree from the School of Physics and Telecommunications Engineering, South China Normal University, Guangzhou, China, in 2021. He is currently working toward the M.Phil. degree with The Hong Kong University of Science and Technology, Hong Kong.

Qingwen Zhang (Graduate Student Member, IEEE) received the M.Phil. degree from The Hong Kong University of Science and Technology, Hong Kong, in 2022. She is currently working toward the Ph.D. degree with the KTH Royal Institute of Technology, Stockholm, Sweden. Her research focuses on dynamic awareness in point clouds.

Bohuan Xue (Member, IEEE) received the B.Eng. degree in computer science and technology from the College of Mobile Telecommunications, Chongqing University of Posts and Telecom, Chongqing, China, in 2018, and the Ph.D. degree from the Department of Computer Science and Engineering, the Hong Kong University of Science and Technology, Hong Kong, China, in 2024. He is currently a Research Fellow with the School of Data Science and Engineering, and Xingzhi College, South China Normal University, Guangzhou, China.

Rui Fan (Senior Member, IEEE) received the B.Eng. degree in automation from the Harbin Institute of Technology in 2015 and the Ph.D. degree in electrical and electronic engineering from the University of Bristol, Bristol, U.K., in 2018. From 2018 to 2020, he was a Research Associate with the Hong Kong University of Science and Technology and Postdoctoral Scholar-Employee with the University of California San Diego between 2020 and 2021. He began his faculty career as a Full Research Professor with the College of Electronics & Information Engineering, Tongji University in 2021. He was promoted to Full Professor in 2022 and attained tenure in 2024, both in the College of Electronics & Information Engineeringe and Shanghai Research Institute for Intelligent Autonomous Systems. His research interests include computer vision, deep learning, and robotics, with a specific focus on humanoid visual perception under the two-streams hypothesis. Prof. Fan served as an Associate Editor for ICRA'23/25 and IROS'23/24, Area Chair of ICIP'24, and Senior Program Committee Member of AAAI'23/24/25/26. He organized several impactful workshops and special sessions in conjunction with WACV'21, ICIP'21/22/23, ICCV'21/25, and ECCV'22. He was honored by being included in the Stanford University List of Top 2% Scientists Worldwide between 2022 and 2025, recognized on the Forbes China List of 100 Outstanding Overseas Returnees in 2023, acknowledged as one of Xiaomi Young Talents in 2023, and awarded the Shanghai Science & Technology 35 Under 35 honor in 2024 as its youngest recipient.

Xiaoyu Tang (Member, IEEE) received the B.S. degree from South China Normal University, Guangzhou, China in 2003, and the M.S. degree from Sun Yat-sen University, Guangzhou, in 2011. He is currently working toward the Ph.D. degree with South China Normal University. He is working with Xingzhi College, South China Normal University, Shanwei, where he is engaged in information system development. His research interests include machine vision, intelligent control, and the Internet of Things. He is a member of the IEEE ICICSP Technical Committee.