

Self-supervised 3D point cloud prediction for moving objects based on deep correlation information

Kai Gao^{1,4}, Yuxiang Zhou¹, Pingping Dong^{2✉}, Lin Hu¹, Rongying Feng¹, Jinlai Zhang¹, Kaihan Zhang³, Inhi Kim³

Cite this article: <https://doi.org/10.26599/JICV.2026.9210103>

ABSTRACT: In autonomous driving, the continuous movement of objects exacerbates point cloud sparsity, making future frame prediction a challenging task. While self-supervised methods excel in such dynamic environments by learning temporal evolution directly from unannotated data, existing approaches often model spatial structures and temporal dynamics independently, limiting their effectiveness in sparse and complex scenes. To overcome this limitation, we reformulate point cloud prediction as a unified spatio-temporal correlation learning problem and propose a novel framework, the Motion-guided Spatio-Temporal Point Cloud Prediction Network (MoCNet). The key idea is to use motion information to guide feature learning. Specifically, we design a dual-branch structure to separately capture spatial structure and temporal variation, and then integrate them into a unified representation. To better model long-range dependencies, we adopt a large-kernel convolution strategy to enhance feature interactions without introducing heavy computational cost. In addition, we employ a dual-frequency representation scheme that decomposes features into high-frequency and low-frequency components. This design helps preserve local details while maintaining global consistency. Experimental results show that the proposed method consistently outperforms existing approaches on multiple benchmark datasets. It achieves higher prediction accuracy and better generalization ability. Moreover, MoCNet supports real-time inference, with an average latency of 20 ms and 15 ms under different settings. This is significantly faster than the typical 3D LiDAR sampling rate. The predicted point clouds are also closer to ground truth in terms of shape, orientation, and spatial position, which improves reliability for downstream autonomous driving tasks.

KEYWORDS: Autonomous driving, prediction, point Cloud, trajectory

1 Introduction

As autonomous driving technology continues to evolve, point cloud prediction has emerged as a crucial component. Utilizing point cloud data to build highly accurate 3D maps of the environment, LiDAR and UAV close-range photogrammetry can be used to directly collect data in a variety of large and complex environments (Luhmann et al., 2023), helping vehicles to understand the precise location and shape of roads, buildings, pedestrians, and other vehicles. Vision and point cloud data can be fused, combining camera and LiDAR data to ensure robust and accurate environmental sensing, bridging the gap between single sensors (Wu et al., 2023). Point cloud prediction combines temporal and spatial information to predict the future trajectories of traffic participants based on their historical behavior and information about their surroundings (Cui et al., 2021). Point cloud data can also be used to predict the state of the road (e.g., slope, curvature, and pavement condition), optimize route planning, and more. Despite the important applications of point cloud prediction in the field of autonomous driving, there are still some shortcomings and challenges in the current research. For example, point cloud data are usually sparse and disordered three-dimensional points (Wang et al., 2024), which are more complex to process compared to two-dimensional images, which makes it difficult to directly apply traditional conventional neural network. Moreover, point cloud prediction in dynamic environments needs to deal with complex object motions and changes, and the prediction accuracy is easily disturbed by dynamic scenes (Zhong et al., 2023).

High-precision point cloud prediction demands significant computational resources. In real-time applications, computational cost and processing speed frequently emerge as key bottlenecks. Furthermore, the manual annotation of point cloud data is inherently cumbersome and time-

consuming, making it exceedingly difficult to obtain high-quality labeled data for large-scale datasets. Li et al. (2021) proposed a novel LiDAR point cloud sequence coding architecture based on clustering and predictive neural networks. For inter-frame point cloud prediction, networks utilizing convolutional LSTM units. Sun et al. (2020) effectively infer future sequential information. However, this approach struggles with disordered and sparse point cloud compression (Cao et al., 2019) and inherently suffers from high algorithmic complexity and slow computational speeds. In addition, Deng and Zakhori (2020) has advanced capabilities in spatio-temporal LiDAR frame prediction for autonomous driving, but suffers from the same limitations of high computational cost, sparse data and sensitivity to sensor noise, making it difficult to achieve the desired results. Based on the above problems, this paper proposes MoCNet based on MoNet (Lu et al., 2021), which is able to predict future point cloud sequences given the past point cloud sequences.

MoCNet employs a self-supervised learning strategy that fundamentally eliminates the reliance on costly manual annotations. Instead of using human-labeled ground truth, the model leverages the inherent spatio-temporal continuity of unlabeled point cloud sequences to construct its own supervisory signals (Fei et al., 2023). Specifically, given a sequence of past LiDAR scans projected into 2D range images, the network aims to predict the future range images and their corresponding valid point masks. The core of this self-supervision lies in utilizing the actual, naturally recorded future LiDAR frames as the absolute ground truth. The supervisory signal is enforced through a composite loss function: a range image loss minimizes the pixel-wise depth differences between the predicted and actual future 2D range images, while a masking loss aligns the predicted point validity masks with the true distribution of valid points. Furthermore, the predicted range images are back-projected into 3D

¹ School of Mechanical and Vehicle Engineering, Changsha University of Science and Technology, Changsha, 410114, China. ² College of Information Science and Engineering, Hunan Normal University, Changsha 410081, China. ³ Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Republic of Korea. ⁴ Research Institute for Innovative Development of Transportation and Energy Integration, Changsha 410114, China.

✉ Corresponding author. E-mail: ppdong@hunnu.edu.cn

Received: August 14, 2026; Revised: September 3, 2026; Accepted: September 19, 2026

© The Author(s) 2026. This is an open access article under the terms of the Creative Commons Attribution 4.0 International License (CC BY 4.0, <http://creativecommons.org/licenses/by/4.0/>).

space, where a chamfer distance loss calculates the exact geometric deviation against the raw, unannotated 3D point clouds at that future timestamp. This multi-faceted self-prediction paradigm forces the network to implicitly learn the underlying dynamic physical laws of moving objects purely from the raw sensor stream, completely avoiding manual annotation dependence.

In MoCNet, the historical 3D point clouds are first projected into 2D range images using spherical projection (Rao et al., 2019), capturing both the spatial coordinates and the depth values of the points. Secondly, the Spatial-Context Content Encoding Stream and the Temporal-Dynamics Motion Encoding Stream are introduced to process these range images. Specifically, the content encoding stream extracts intra-frame geometric correlations using Farthest Point Sampling (FPS), k-Nearest Neighbors (k-NN) search and feature concatenation (Abbasifard et al. 2014). Meanwhile, the motion encoding stream performs differential computations and concatenates features across adjacent time steps to extract transient temporal dynamics. This dual-stream decoupling paradigm effectively captures both static spatial structures and dynamic kinematic changes, thereby significantly improving prediction accuracy.

Subsequently, the input range images, spatial-context features, and temporal-dynamics motion features are concatenated along the channel dimension. This fused 3D tensor is then fed into a dual-frequency spatiotemporal pipeline to process the comprehensive features derived from the past P-frame sequences. Within this architecture, the OctSTA decoder (Chen et al., 2019) incorporates multiple Spatio-Temporal Attention (STA) blocks. These blocks utilize large-kernel convolutions to implicitly simulate the attention mechanism (Guo et al., 2022), efficiently extracting long-range spatio-temporal dependencies from the input representations to output the predicted range images for the future F-frames.

The final output tensor comprises two channels: one predicting the future range images, and the other predicting the validity probability of each point. The predicted validity probability is thresholded at 0.5 to obtain a binary validity mask. This threshold is adopted as a practical decision boundary. Lowering the threshold may improve point-cloud completeness but introduce more noise, whereas increasing it provides more conservative filtering but may remove valid points. Finally, these predicted range images are separated along the temporal dimension and back-projected to reconstruct the predicted 3D point cloud sequences.

The main contributions of this paper are summarized as follows:

- **Unified Motion-Guided Spatio-Temporal Prediction Framework:** We propose a self-supervised point cloud prediction framework, termed Motion-guided Spatio-Temporal Point Cloud Prediction Network (MoCNet). The framework reformulates point cloud prediction as a unified learning problem, where spatial structures and temporal variations are jointly modeled. By leveraging motion information to guide feature learning, the model can effectively capture both geometric relationships within each frame and dynamic changes across frames without requiring manual annotations.

- **Efficient Long-Range Dependency Modeling via Large-Kernel Convolution:** We develop a lightweight feature interaction strategy based on large-kernel convolution to enhance long-range dependency modeling. This design enables the network to focus on informative regions and suppress irrelevant noise, while avoiding the high computational cost of conventional attention mechanisms.

- **Dual-Frequency Spatio-Temporal Representation Learning:** We introduce a dual-frequency representation scheme that separates features into high-frequency and low-frequency components. This design allows the model to preserve local geometric details while maintaining global structural consistency, improving both representation quality and computational efficiency.

- **Accurate and Real-Time Point Cloud Prediction:** Extensive experiments on the KITTI Odometry and Apollo DaoxiangLake datasets demonstrate that the proposed method consistently outperforms existing approaches. It achieves higher prediction accuracy and better robustness in

long-term forecasting, while maintaining real-time inference with a latency of 15–20 ms (50–67 Hz), significantly faster than typical LiDAR sampling rates.

2 RELATED WORK

To contextualize the aforementioned contributions and highlight the architectural gaps in current literature, this section reviews the evolution of point cloud prediction methods. These existing approaches can be broadly categorized into the following types based on their representational methodologies:

2.1. Occupancy grid based prediction methods

Mahjourian et al. (2022) proposed the Occupancy Flow Fields representation that can be used for motion prediction by multiple agents. The representation is a spatio-temporal grid where each grid cell contains the probability of the cell being occupied by any agent. And a two-dimensional flow vector that represents the direction and magnitude of the motion of the cell. Additionally, the method successfully mitigates the motion prediction in the shortcomings of the two most commonly used representations (trajectory sets and occupancy grids). Khurana et al. (2023) reshaped the prediction task into a spatio-temporal occupancy prediction. Rendering point cloud data from 4D occupancy predictions solves the problem of the high cost of obtaining ground-truthed 4D occupancies. Consequently, the prediction task was expanded from an initial bird's-eye view to a comprehensive full three-dimensional perspective.

2.2. Prediction method based on point cloud sequences

Ding et al. (2024) proposed Meteor Module, a deep learning method for dynamic 3D point cloud sequences, which processes point cloud sequences and extracts spatio-temporally localized features. The method estimates the position of the point in frame t in frames $t-1$, $t-2$, etc. by means of scene flow, constructs a neighborhood centered on the estimated position of each frame with a radius, and aggregates the information in these neighborhoods to learn the features of each point. Liu et al. (2019) demonstrates better performance than previous grid-based approaches. Fan and Yang (2019) combining Recurrent Neural Networks (RNN) and point cloud processing techniques for processing dynamic point cloud data. It realizes temporal prediction of point cloud sequences, solves the problem of capturing temporal dependencies in point cloud sequences, and demonstrates superior temporal prediction performance on multiple dynamic point cloud datasets.

Rui et al. (2023) proposed a hierarchical feature extraction method combining local and global features to process point cloud data. It improves the accuracy and efficiency of point cloud data processing through a hierarchical feature learning approach. The model demonstrates superior performance on multiple 3D point cloud datasets and is widely used in point cloud classification, segmentation and dynamic prediction tasks. Liu et al. (2019) proposed FlowNet3D, which is capable of learning scene flow from 3D point clouds and realizing prediction of dynamic point clouds through point feature extraction and scene flow estimation modules. This is the first time that scene flow learning has been introduced in a 3D point cloud, enabling the capture of dynamic changes in the point cloud.

2.3. Approach based on self-supervised learning

Sheng et al. (2023) proposed a self-supervised paradigm for point cloud sequence understanding, designing two tasks, contrast prediction and reconstruction, to synergistically learn integrated spatio-temporal representations. Dense point cloud segments are fed into the encoder to extract embeddings, while the context-aware autoregressor is used to make predictions about the last target segment. Local and global comparisons are



learned between predictions and targets, and the decoder uses the predictions to reconstruct the original point cloud sequence. Building upon static masked point modeling, the most recent advancements have expanded self-supervised paradigms directly to dynamic 4D point cloud sequences. For instance, Yuan et al. (2025) proposed a self-supervised 4D scene reconstruction technique for egocentric videos to capture the highly non-linear kinematics and temporal evolutions of dynamic environments. Furthermore, to address the immense computational burden of processing dense 4D point cloud videos, Liu et al. (2025) introduced Mamba4D, a cutting-edge framework that leverages disentangled spatial-temporal state space models for highly efficient point cloud video understanding. Despite these significant strides in efficiency and self-supervised learning, achieving a strict balance between long-term geometric prediction fidelity (i.e.,

mitigating temporal error accumulation) and real-time inference speed on edge devices remains a formidable challenge.

3 METHODOLOGY

While the aforementioned grid-based and sequence-based methods have advanced the field, they frequently struggle to explicitly balance fine-grained spatial correlations with efficient temporal dynamic modeling. Self-supervised paradigms mitigate the annotation bottleneck but often incur prohibitive computational costs. Addressing these specific limitations forms the foundation of our proposed architecture.

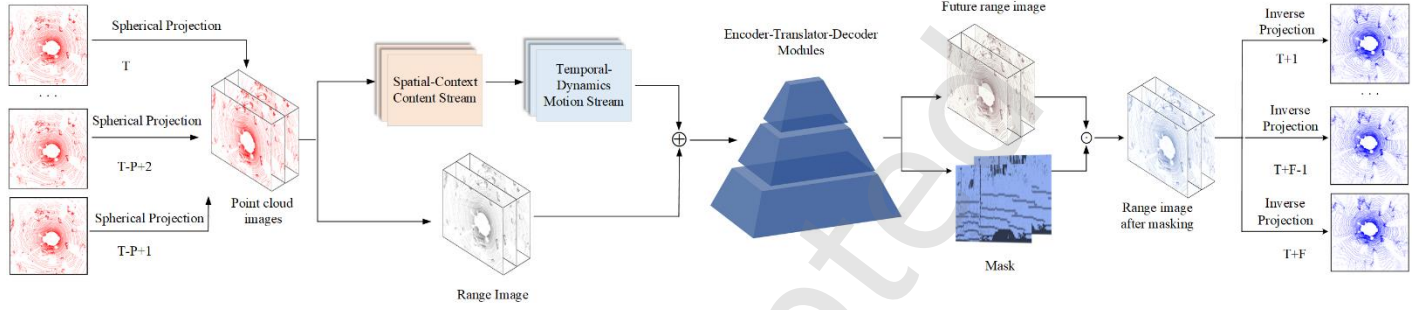


Fig. 1: MoCNet General Framework Diagram. First, the historical 3D point cloud sequence is mapped onto a 2D plane via spherical projection to generate 2D range images. Subsequently, these range images are fed into the Spatial-Context Content Encoding Stream and the Temporal-Dynamics Motion Encoding Stream. These streams respectively extract the intra-frame spatial features and the inter-frame temporal motion features. Next, the extracted content and motion features are concatenated to form a 3D tensor. This tensor is then processed by the encoder-OctSTA-decoder architecture. Through multi-level feature extraction and fusion, this architecture captures complex spatio-temporal dependencies and motion patterns across different time steps. Ultimately, the model outputs a predicted future range image. This image is converted back into a 3D point cloud via inverse projection, successfully accomplishing the prediction of the future point cloud state.

TABLE I: Summary of Key Mathematical Notations

Symbol	Description
P, F	Number of historical input frames and predicted future frames.
H, W	Spatial resolution (height and width) of 2D range images.
B	Batch size of the input tensor.
N, K	Number of key points (FPS) and nearest neighbors (k-NN).
C	Number of feature channels.
α	Channel ratio allocated to the low-frequency pathway in the Octave Convolution.
Content & Motion Encoding	
P_N, P_{NK}	Sampled key points tensor and local neighborhood point set.
C_P^t	Spatial content correlation features at time step t .
CPS, P_{NS}, P_{DS}	Correlation, coordinate, and depth feature differentials between adjacent frames.
E, M_P	Concatenated spatio-temporal tensor and fused motion-aware feature map.
OctSTA Decoder	
X_H, X_L	High-frequency and low-frequency input feature representations.
X_{DH}, X_{DL}	Intermediate spatial features extracted via 2D Octave convolutions.
X_{DdH}, X_{DdL}	Refined spatial features processed through Octave dilated convolutions.
X_{AH}, X_{AL}	The final attention-enhanced high-frequency and low-frequency output features.
Prediction & Loss functions	
$r_{ij}, \hat{r}_{ij}$	Ground truth and predicted depth values at pixel coordinates (i, j) .

$\gamma_{ij}, \hat{m}_{ij}$	Ground truth validity (binary) and predicted probability of a valid point at pixel (i, j) .
$S_t, \hat{S}_t$	Ground truth and back-projected predicted 3D point cloud at future frame t .
$L_{R,t}, L_{M,t}, L_{CD,t}$	Range image loss, masking loss, and Chamfer Distance loss at future frame t .

To address the issue of weak inter-point connections in point clouds, this paper introduces an enhanced point cloud prediction algorithm named MoCNet, as shown in Fig. 1. MoCNet is designed to predict future point cloud sequences based on past point cloud data. The process begins with the projection of the historical 3D point cloud onto a 2D image using spherical projection (Mersch et al., 2022), which captures both the coordinate and distance values of the point cloud. To explicitly highlight the structural advancements of MoCNet over existing baselines, Tab. I summarizes the key architectural differences. Unlike MoNet and TCNet which rely on single-stream extractions and standard convolutions, MoCNet integrates a dual-branch encoding mechanism and an attention-simulating OctSTA decoder to balance accuracy and computational load.

The proposed dual-branch encoding mechanism consists of two parallel components: the Spatial-Context Content Encoding Stream and the Temporal-Dynamics Motion Encoding Stream. The Spatial-Context Content Encoding Stream extracts geometric correlations within the point cloud at a single time step. Meanwhile, the Temporal-Dynamics Motion Encoding Stream captures dynamic changes across different time steps by analyzing variations in the point cloud representations. Subsequently, the depth, correlation, and motion features are fused into a 3D tensor. The encoder-translator-decoder architecture then processes this tensor to extract comprehensive spatio-temporal features. Specifically, the translator utilizes large-kernel convolutions to approximate global dependencies efficiently, while the entire encoder-decoder pipeline adopts a dual-frequency scheme to balance representation fidelity and computational load. Finally, the network outputs a future range image, which is back-projected to generate the predicted 3D point cloud.

To facilitate a clear presentation of the proposed MoCNet architecture and its mathematical formulations, the principal mathematical symbols and variables used throughout this paper are summarized in Tab. I.

3.1. Spatial-Context Content Encoding Stream

The fundamental rationale for designing a dedicated content encoder based on FPS and k-NN lies in the inherent sparsity and unstructured nature of raw LiDAR data. Processing massive, unordered point clouds via standard grid-based convolutions inevitably leads to severe computational redundancy, largely due to operations on empty voxel spaces. Within the Spatial-Context Content Encoding Stream, the current frame point cloud undergoes a hierarchical feature extraction process to capture both local and global geometric information as shown in Fig. 2. Initially, FPS is applied to select N representative key points $P_N \in \mathbb{R}^{B \times 3 \times N}$ from the input point cloud. This uniform sampling strategy ensures that the key points are well-distributed, effectively preserving the underlying topological structure while minimizing computational redundancy. Subsequently, the k-NN algorithm is employed to establish local neighborhoods around each key point, generating $N \times K$ local points $P_{NK} \in \mathbb{R}^{B \times 3 \times N \times K}$. This explicit neighborhood construction enables the stream to accurately capture fine-grained spatial relationships, which are essential for robust downstream feature learning.

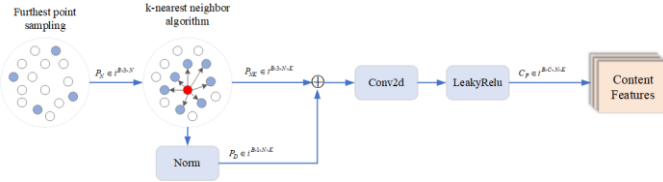


Fig. 2: Structure of the Spatial-Context Content Encoding Stream. The Spatial-Context Content Encoding Stream adopts a hierarchical feature extraction strategy to extract N representative key points from the input point cloud via FPS. It then constructs a local neighborhood point set for each key point using the k-NN algorithm. Subsequently, the fused features are fed into a multilayer convolutional network, where local correlation features are extracted through pointwise convolutional operations. The final output is a high-dimensional feature matrix.

For each neighboring point q_j , its depth feature $d_j = \|q_j\|_2$ is computed, representing the Euclidean distance from the origin. This depth information is concatenated with the Cartesian coordinates (x_j, y_j, z_j) , forming the enhanced feature vectors $f_j = [x_j, y_j, z_j, d_j] \in \mathbb{R}^4$. These concatenated feature vectors are then passed through a 2D convolutional layer with C kernels. This operation generates the spatial correlation tensor $C_p \in \mathbb{R}^{B \times C \times N \times K}$, explicitly capturing the fine-grained inter-point relationships within the point cloud. Ultimately, this hierarchical extraction and feature fusion approach enables the stream to effectively model complex spatial dependencies, thereby significantly improving the overall predictive performance of the network.

3.2. Temporal-Dynamics Motion Encoding Stream

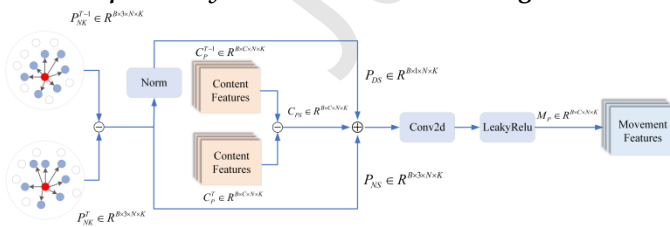


Fig. 3: Structure of the Temporal-Dynamics Motion Encoding Stream. The inputs to this stream comprise the point cloud coordinates a and spatial-context content features b from the previous time step, along with the coordinates c and content features d from the current time step. Initially, the content features are normalized. Subsequently, the concatenated features undergo a nonlinear transformation via a 2D convolutional layer (Conv2D) and a LeakyReLU activation function to extract the final temporal-dynamics motion features.

While the spatial-context content encoding stream successfully captures the static intra-frame geometry, extracting accurate kinematic features from dynamic scenes remains a formidable challenge. Conventional

approaches typically rely on 3D convolutions or recurrent units (e.g., Long Short-Term Memory (LSTM) networks) to implicitly learn temporal dynamics. However, these black-box methods often struggle to maintain geometric fidelity over extended horizons, leading to severe error accumulation. The theoretical rationale behind our explicitly designed temporal-dynamics motion encoding stream is to introduce a strict physics-based inductive bias into the network as shown in Fig. 3.

Instead of leaving motion estimation to implicit feature fusion, the stream mathematically enforces the computation of coordinate and semantic differentials across adjacent time steps. This explicit temporal differencing directly proxies the physical velocity and acceleration of moving targets. By structurally separating these dynamic motion features from the static background representations, the network achieves highly robust temporal modeling, effectively preventing the spatial deformation and recursive drift that plague standard recurrent architectures. Let t and $t+1$ denote two consecutive time steps, with their content-encoded features represented as $C_p^t \in \mathbb{R}^{B \times C \times N \times K}$ and $C_p^{t+1} \in \mathbb{R}^{B \times C \times N \times K}$, respectively. The motion features are derived through several key steps to capture temporal dynamics and motion-related patterns within the point cloud sequence. These steps are crucial for understanding the changes between consecutive time steps, which is vital for tasks like motion tracking or dynamic scene reconstruction.

First, the motion features are obtained by performing feature differencing. The correlation feature differential is computed as:

$$C_{PS} = C_p^{t+1} - C_p^t \in \mathbb{R}^{B \times C \times N \times K}$$

Where C_p^t and C_p^{t+1} denote the spatial correlation features at time steps t and $t+1$, respectively. This differential captures the temporal variations in the spatial relationships between points in the cloud. Similarly, the coordinate differential is calculated as:

$$P_{NS} = P_N^{t+1} - P_N^t \in \mathbb{R}^{B \times 3 \times N \times K}$$

Where P_N^t and P_N^{t+1} denote the sampled key-point tensors at the two consecutive time steps. This represents the differences in the positions of points across consecutive frames, helping the model capture the movement of the points in space. The next step involves depth feature extraction, where the Euclidean norm of the coordinate differentials is computed to quantify the magnitude of motion. This is denoted as follows,

$$P_{DS} = \|P_{NS}\|_2 \in \mathbb{R}^{B \times 1 \times N \times K}$$

Where $\|\cdot\|_2$ denotes the Euclidean norm, and P_{DS} represents the magnitude of the inter-frame coordinate displacement. The depth features provide a scalar representation of motion magnitude, enabling the model to focus on how much motion occurred at each point. Following this, the depth features P_{DS} , correlation differentials C_{PS} , and coordinate differentials P_{NS} are concatenated along the channel dimension to form a comprehensive feature tensor:

$$E = [C_{PS}; P_{NS}; P_{DS}] \in \mathbb{R}^{B \times (C+4) \times N \times K}$$

This concatenation fuses the temporal motion information with the spatial and depth-related features, yielding a comprehensive tensor that encapsulates rich representations of the temporal dynamics within the point cloud. Subsequently, the fused tensor E is processed by a 2D convolutional layer with C kernels to generate the final motion-aware features. These features provide the essential kinematic cues required for dynamic scene understanding and accurate future frame prediction:

$$M_p = \text{Conv}(E) \in \mathbb{R}^{B \times C \times N \times K}$$

The MoCNet framework processes the input feature tensor $X \in \mathbb{R}^{N \times 6 \times P \times H \times W}$, where N is the batch size, 6 represents the fused features, P is the number of temporal frames, and $H \times W$ is the spatial resolution as shown in Fig. 4. The processing is performed through several stages that progressively refine the feature representations, making them more suitable for understanding both spatial and temporal dependencies.

First, a masking operation is applied to filter invalid points, generating a masked tensor $\tilde{X}$. The 3D Octave Convolution Encoder then

employs dual-frequency processing to handle both high-frequency and low-frequency components of the data. The high-frequency path, which captures fine-grained details, processes the data using $(1 - \alpha)C$ kernels:

$$X_H^0 \in \mathbb{R}^{N \times (1-\alpha)C \times P \times H \times W}$$

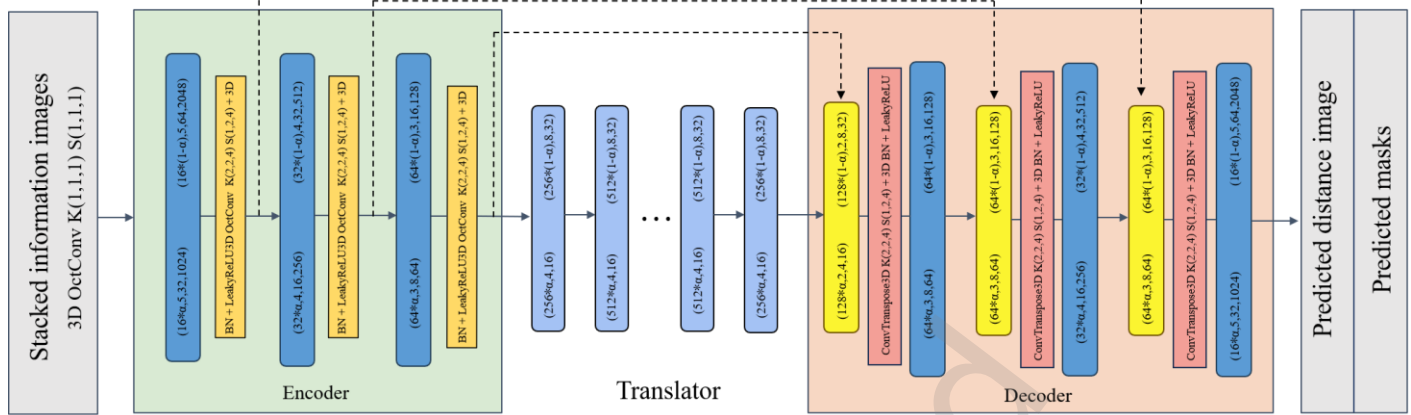


Fig. 4: The figure illustrates the overall structure of MoCNet, where solid lines represent information flows and dashed lines indicate skip connections. Each colored box provides detailed information about the specific components that constitute the network. In this paper, we propose an architecture based on 3D OcConv and Core Transport for processing stacked information images and generating predicted distance images and masks. The encoder part comprises multiple 3D convolutional layers, each followed by Batch Normalization and LeakyReLU activation functions, to progressively extract high-dimensional features. The decoder mirrors the encoder's structure, and the final network outputs the predicted distance image and mask.

The low-frequency path, designed to capture global patterns, processes the data using αC kernels, but at half the spatial resolution:

$$X_L^0 \in \mathbb{R}^{N \times \alpha C \times P \times \frac{H}{2} \times \frac{W}{2}}$$

In the spatio-temporal decoding stage, the high-frequency and low-frequency tensors are transformed into four-dimensional representations. The high-frequency tensor is represented as:

$$\tilde{X}_H^i \in \mathbb{R}^{N \times (1-\alpha)C_i \times (P_i - t_i) \times \frac{H_i}{h_i} \times \frac{W_i}{w_i}}$$

The low-frequency tensor is represented as:

$$\tilde{X}_L^i \in \mathbb{R}^{N \times \alpha C_{i+1} \times (P_i - t_i) \times \frac{H_i}{2h_i} \times \frac{W_i}{2w_i}}$$

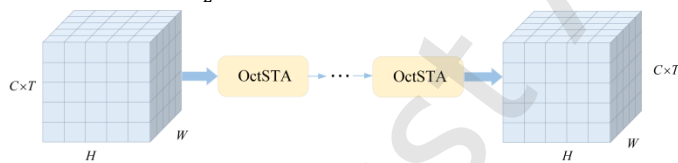


Fig. 5: OctSTA translator structure. The OctSTA translator contains several Spatio-Temporal Attention (STA) blocks, which use large kernel convolution to model the attention mechanism and extract spatio-temporal attention features from the input representations.

These tensors are then processed through 2D Octave convolutions to capture spatio-temporal dependencies, as illustrated in Fig. 5. The final output tensors effectively capture both spatial and temporal features, allowing for a deep understanding of the dynamic patterns in the point cloud sequence.

By restricting the low-frequency computations to this halved spatial dimension ($H/2$ and $W/2$), the overall spatial resolution is reduced by a factor of 4 ($1/2 \times 1/2$). This mechanism drastically significantly lowers the floating-point operations (FLOPs) required for standard 3D convolutions. Since low-frequency components typically contain redundant spatial information (e.g., static background structures), this spatial reduction eliminates unnecessary computations without discarding the essential global contextual structure, thus maintaining high predictive performance while accelerating inference. The downsampled tensors are represented as:

$$X_H^i \in \mathbb{R}^{N \times (1-\alpha)C_i \times (P_{i-1} - t_{i-1}) \times \frac{H_{i-1}}{h_i} \times \frac{W_{i-1}}{w_i}}$$

3.3. OctSTA Decoder and STA Block

The OctSTA decoder incorporates several STA blocks. These blocks are explicitly designed to capture complex temporal dependencies and global spatial contexts across different frequency components. Standard self-attention mechanisms (e.g., Vision Transformers) excel at this global modeling. However, their quadratic computational complexity, $O(N^2)$, is functionally prohibitive for dense 4D point cloud sequences.

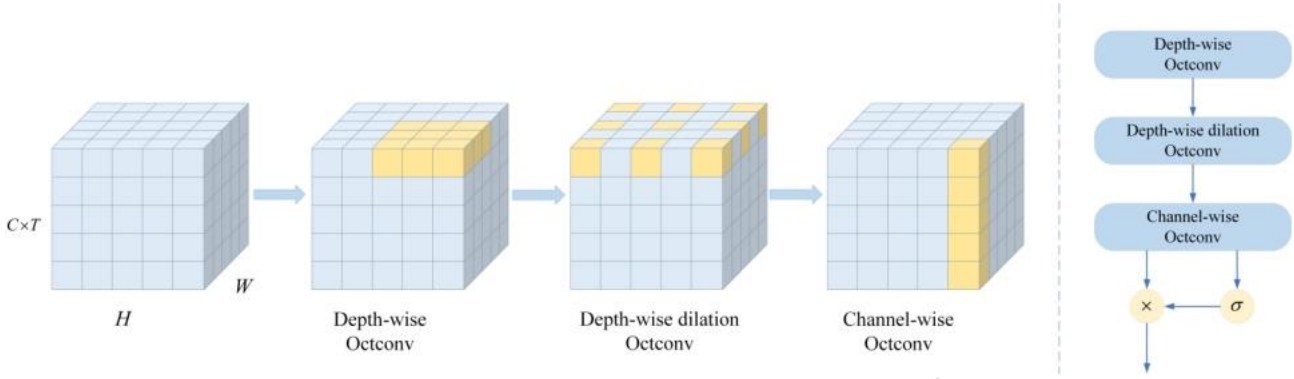


Fig. 6: Structure of the STA Block. This spatio-temporal attention block leverages Depth-wise Convolutions and OctConv to capture complex spatio-temporal dependencies within point cloud data. Initially, the block extracts local features across the spatial (H, W) and channel (C) dimensions via a Depth-wise OctConv. Subsequently, a Depth-wise Dilated OctConv is employed to expand the effective receptive field, enabling the capture of broader spatio-temporal dependencies. Finally, a Channel-wise OctConv adaptively weights the feature maps along the channel dimension. This critical operation highlights informative features while effectively suppressing redundant noise.

To overcome this bottleneck, the STA block approximates attention mechanisms using large-kernel and dilated convolutions as shown in Fig. 6. Recent theoretical advancements demonstrate that large-kernel convolutions effectively expand the network's Effective Receptive Field (ERF). This expansion allows them to implicitly mimic the long-range dependency extraction of spatial attention. Consequently, the STA block achieves Transformer-like global contextual awareness. Simultaneously, it preserves the translation equivariance and linear computational complexity, $O(N)$, of traditional convolutions.

Building upon this efficient foundation, the STA block first processes the high-frequency (X_{Hi}) and low-frequency (X_{Li}) components. It applies a 2D Octave convolutional layer with a kernel size of $(2d-1) \times (2d-1)$ along the depth dimension. This step generates the intermediate feature maps X_{DH} and X_{DL} . By operating on both frequency components, the model better captures fine-grained details alongside broader, low-frequency information.

Next, a 2D Octave dilated convolution is applied to X_{DH} and X_{DL} along the depth dimension. This layer utilizes a kernel size of $\frac{K}{d} \times \frac{K}{d}$ and an expansion coefficient of d . This operation produces the refined feature maps X_{DdH} and X_{DdL} . The dilated convolution further increases the receptive field without reducing spatial resolution. Thus, it enables the model to capture even longer-range dependencies and global patterns.

Following this, a 1×1 2D Octave convolutional layer processes these refined maps along the channel dimension. This generates compact feature representations, X_{CH} and X_{CL} . These maps are then split into two equal halves. The first half passes through a Sigmoid activation function to yield X_{CSH} and X_{CSL} . The second half (X_{CGH} and X_{CGL}) is then multiplied element-wise by these activated components. This multiplication results in the final attention-enhanced outputs, X_{AH} and X_{AL} .

Ultimately, these attention-enhanced features significantly refine the model's predictive capabilities. The STA block effectively learns where to attend in both spatial and temporal domains. Through this efficient large-kernel approximation, the model improves its performance in motion prediction and range image generation while maintaining real-time inference speeds.

The computational process of the STA block can be formulated as Eq. (1):

$$X_{DH}, X_{DL} = \text{OctaveConv}2D_{(2d-1) \times (2d-1)}(X_{hi}, X_{li}) \quad (1)$$

where X_{hi} and X_{li} denote the input high-frequency and low-frequency feature

maps, respectively. The function $\text{OctaveConv}2D$ represents the two-dimensional Octave convolution operation with a kernel size of $(2d-1) \times (2d-1)$. This operation extracts hierarchical spatial features, generating the high-frequency output X_{DH} and low-frequency output X_{DL} .

$$X_{DH}, X_{DL} = \text{OctaveConv}2D_{\frac{d}{2} \times \frac{d}{2}}(X_{DH}, X_{DL}) \quad (2)$$

In this step, X_{DH} and X_{DL} are further processed through another Octave convolution with a kernel size of $\frac{k}{d} \times \frac{k}{d}$, yielding the refined high-frequency feature X_{DdH} and low-frequency feature X_{DdL} .

$$X_{CH}, X_{CL} = \text{OctaveConv}2D_{1 \times 1}(X_{DdH}, X_{DdL}) \quad (3)$$

Here, a 1×1 Octave convolution is applied to X_{DdH} and X_{DdL} , producing compact and transformed feature representations: X_{CH} (high-frequency output) and X_{CL} (low-frequency output).

$$X_{CSH}, X_{CGH} = \text{split}(X_{CH}) \quad (4)$$

The function $\text{split}(X_{CH})$ divides the high-frequency feature X_{CH} into two separate components based on the number of channels: X_{CSH} (the first subset of high-frequency features) and X_{CGH} (the second subset of high-frequency features).

$$X_{CSL}, X_{CGL} = \text{split}(X_{CL}) \quad (5)$$

Similarly, the low-frequency feature X_{CL} is split into two parts: X_{CSL} (the first subset of low-frequency features) and X_{CGL} (the second subset of low-frequency features).

$$X_{AH} = \sigma(X_{CGH}) \odot X_{CSH} \quad (6)$$

where $\sigma(X_{CGH})$ denotes the activation of the high-frequency feature X_{CGH} using the Sigmoid function. The element-wise multiplication ($\odot$) between $\sigma(X_{CGH})$ and X_{CSH} produces the final high-frequency attention feature X_{AH} .

$$X_{AL} = \sigma(X_{CGL}) \odot X_{CSL} \quad (7)$$

where $\sigma(X_{CGL})$ represents the activation of the low-frequency feature X_{CGL} using the Sigmoid function, and the resulting element-wise multiplication yields the final low-frequency attention feature X_{AL} .

Finally, the decoded feature tensor is transformed and fed into the decoder, which consists of ConvTranspose3D and 3D Octave convolution

layers. This decoding process reconstructs a feature tensor of size $(N, 2, F, H, W)$. The decoder is designed as a mirror image of the encoder and incorporates skip connections, allowing information to flow from the decoder back to the encoder. These skip connections help the encoder capture more meaningful features, enhancing inference performance.

In the encoder-translator-decoder architecture, the number of predicted future range images is fixed. However, autoregressive prediction can be achieved by recursively using the predicted range images as input tensors, enabling long-term forecasting.

The output feature tensor consists of two channels: The first channel represents the predicted future range image. The second channel encodes the probability that each point in the range image is valid.

By filtering out points with a probability lower than 0.5, a refined range image containing only valid points is obtained.

4 Experiments

Having established the theoretical framework and the spatio-temporal decoding mechanisms of MoCNet, this section presents a comprehensive empirical evaluation designed to validate its efficacy against state-of-the-art baselines. The experimental validation in this study was conducted on a high-performance computing server equipped with an Intel(R) Xeon(R) CPU E5-2698 v3@ 2.30GHz and a Tesla P40 GPU (24GB). The proposed self-supervised neural network, MoCNet, which integrates a Temporal-Dynamics Motion Encoding Stream, a Spatial-Context Content Encoding Stream, and a dual-frequency spatiotemporal pipeline, demonstrates superior capability in extracting spatiotemporal information from input past point cloud sequences and accurately predicting future point cloud sequences.

Through comprehensive evaluation on three distinct datasets, the experimental results reveal that our method significantly outperforms existing point cloud prediction approaches in terms of prediction accuracy, as evidenced by superior performance metrics in both Chamfer Distance and Earth Mover's Distance across different test sets. Furthermore, the proposed method exhibits strong generalization capability in unknown environments and achieves real-time prediction performance that surpasses the frame rate of conventional 3D LiDAR sensors (10 Hz). To validate the contribution of each component, we conducted systematic ablation studies to analyze the individual effects of the content encoding stream, the motion encoding stream, the OctSTA decoder, and the OctConv mechanism on the overall model performance, providing valuable insights into the architectural design and its impact on prediction quality.

Algorithm 1 Overall framework of our method

Require: Point cloud sequence from LiDAR: $P=[P_0, \dots, P_n]$

Ensure: Future point clouds: $F=[F_0, \dots, F_n]$

- 1: **Step 1 (Range-image projection & content encoding):**
- 2: **for** $i = 0, \dots, n$ **do**
- 3: $R_i^p, r_i^p \leftarrow \text{proj}(P_i)$ ▢ 2D projection and range-image
- 4: $C_i \leftarrow \text{projContent}(R_i^p)$ ▢ content encoding from range image
- 5: **end for**
- 6: Let $rmList \leftarrow [M_0^p, \dots, M_n^p]$ and $rclist \leftarrow [C_0^p, \dots, C_n^p]$.
- 7: **Step 2 (Motion encoding):**
- 8: **for** $i = 0, \dots, n-1$ **do**
- 9: $M_i^p \leftarrow \text{Motion}(C_i^p, C_{i+1}^p)$
- 10: **end for**
- 11: $M_n^p \leftarrow \text{Motion}(C_n^p, C_n^p)$ ▢ boundary case
- 12: Let $rmList \leftarrow [M_0^p, \dots, M_n^p]$.
- 13: **Step 3 (Encoder - 3D OctConv stack):**

- 14: $O^p \leftarrow \text{concat}(rplist, rclist, rmList)$
 - 15: **for** $i = 0, \dots, m$ **do**
 - 16: **if** $i = 0$ **then**
 - 17: $(X_0^H, X_0^L) \leftarrow 3D_OctConv(O^p)$
 - 18: **else**
 - 19: $(X_i^H, X_i^L) \leftarrow 3D_OctConv(X_{i-1}^H, X_{i-1}^L)$
 - 20: **end if**
 - 21: **end for**
 - 22: Encoder output: (X_m^H, X_m^L) .
 - 23: **Step 4 (Spatio-temporal attention /translator):**
 - 24: $(\tilde{X}^H, \tilde{X}^L) \leftarrow (X_m^H, X_m^L)$
 - 25: **for** $j = 1, \dots, k$ **do**
 - 26: $(\tilde{X}^H, \tilde{X}^L) \leftarrow STA(\tilde{X}^H, \tilde{X}^L)$
 - 27: **end for**
 - 28: **Step 5 (Decoder - upsampling & reconstruction):**
 - 29: $Y \leftarrow \tilde{X}$
 - 30: **for** $i = m, \dots, 0$ **do**
 - 31: $Y \leftarrow \text{ConvTranspose3D}(Y)$
 - 32: $Y \leftarrow 3D_OctConv(Y)$
 - 33: **end for**
 - 34: Decoded feature tensor: $M^F \leftarrow Y$.
 - 35: **Step 6 (Reprojection to point cloud):**
 - 36: **for** $i = 0, \dots, n$ **do**
 - 37: $F_i \leftarrow \text{reproj}(M^F, rplist[i])$ ▢ use decoded features + corresponding range-image to reconstruct point cloud
 - 38: **end for**
 - 39: **return** $F = [F_0, \dots, F_n]$
-

4.1 Loss function

MoCNet has multiple losses, including range image loss, mask loss, and chamfer distance loss (Lin et al., 2023), where the range image loss function is shown as Eq. (8):

$$L_{R,t} = \frac{1}{HW} \sum_{i,j} \Delta r_{i,j}$$

$$, \text{ with } \Delta r_{i,j} = \begin{cases} \|\hat{r}_{i,j} - r_{i,j}\|_1, & \text{if } r_{i,j} > 0 \\ 0, & \text{otherwise} \end{cases} \quad (8)$$

where the average range image loss at future frame t is the predicted range image $(\hat{r}_{i,j})_t \in \mathbb{R}^{HW}$ and the true range image $(r_{i,j})_t \in \mathbb{R}^{HW}$ computed by the L1 loss function. The loss function of the mask is shown as Eq. (9):

$$(r_{i,j})_t \in \mathbb{R}^{HW}, L_{M,t} = \quad (9)$$

$$\frac{1}{HW} \sum_{i,j} -y_{i,j} \log(\hat{m}_{i,j}) - (1 - y_{i,j}) \log(1 - \hat{m}_{i,j})$$

The average mask loss $L_{M,t}$ of the future frame t is the predicted probability that a point in the range image is a valid point $(\hat{m}_{i,j})_t \in \mathbb{R}^{HW}$ and the value of whether a point in the real range image is a valid point $y_{i,j}$ is calculated by the BCEWithLogitsLoss loss function, where $y_{i,j}$ has only two values, 0 and 1.0 represents invalid points and 1 represents valid points.

The chamfer distance loss function is shown as Eq.(10):

$$L_{CD,t} = \frac{1}{N} \sum_{\hat{p} \in \hat{S}_t} \min_{p \in S_t} \|\hat{p} - p\|^2 + \frac{1}{M} \sum_{\hat{p} \in \hat{S}_t} \min_{p \in S_t} \|\hat{p} - p\|^2 \quad (10)$$

Both the range image loss and the masking loss consider only the predicted 2D range image and do not consider the back-projected 3D point cloud. For comparison with the 3D point cloud based methods, the chamfered distance loss $L_{CD,t}$ can be computed by back-projecting the 3D point cloud $\hat{S}_t = \hat{p} \in \mathbb{R}^3, |\hat{S}_t| = N$ by the masked range image containing only valid points,

at frame t in the future. with the real point cloud $S_t = p \in R^3$, $|S_t|=M$, is computed. For the time $t=T$ at which we are currently located, the average loss L for the future F frames is shown as Eq. 11:

$$L = \frac{1}{F} \sum_{t=T+1}^{T+F} L_{R,t} + \alpha_M L_{M,t} + \alpha_{CD} L_{CD,t} \quad (11)$$

Although EMD is used as an evaluation metric, it is not included in the training objective, with $\lambda_{EMD} = 0$. Chamfer Distance is retained for geometric supervision because it is more practical for repeated optimization. Incorporating EMD would introduce additional point-set matching computations and increase the training cost, particularly in the multi-frame prediction setting. Because the Chamfer distance calculation involves computing pairwise distances between all points in the back-projected and ground-truth point clouds, it introduces significant computational overhead. Consequently, optimizing solely with the 2D image-based loss functions ($L_{R,t}$ and $L_{M,t}$) is approximately 2.5 times faster than utilizing the combined loss ($L_{R,t}$, $L_{M,t}$, and $L_{CD,t}$). Therefore, $L_{R,t}$ and $L_{M,t}$ are used in TCNet to pre-train the network first, followed by adding $L_{CD,t}$ to fine-tune the trained model.

4.2 Projected results

The experimental validation in this study is conducted based on the KITTI Odometry dataset, which contains 22 continuous sequences. We use sequences 0-5 as the training set, sequences 6-7 as the validation set, and sequences 8-10 as the test set. During data preprocessing, the raw 3D point clouds are projected into 2D range images with a fixed high spatial resolution of $H=64$ and $W=2048$. These projections precisely capture a vertical field of view (FOV) ranging from -24.8° to 2.0° and a depth range spanning from 1.0m to 80.0m. Each sequence is divided into sample units consisting of P frames of historical range images and F frames of future range images, with adjacent samples spaced 1 frame apart to maintain temporal continuity. For the spatial-context content encoding stream, each input point cloud is represented by 65,536 points. FPS is employed to select 2,048 representative key points, and k-NN retrieves 64 neighboring points from the

original point cloud for each key point. The resulting local neighborhoods are subsequently processed for geometric feature extraction.

Specifically, following the configuration of mainstream prediction tasks, the temporal window is explicitly set to take $P=5$ historical frames as input to forecast the subsequent $F=5$ future frames. The network training employs an initial learning rate of 0.001, and the learning rate is dynamically adjusted using an exponential decay strategy (decay factor of 0.99). The training process utilizes a batch size of 6, distributed across a computing cluster of 5 GPUs, with the Octave convolution ratio parameter α configured to 0.125 for both input and output channels. Crucially, to ensure a strictly fair and rigorous evaluation, we explicitly declare that all baseline methods evaluated in this study (including MoNet, TCNet, and TSMNet) were trained from scratch and evaluated under the exact same configurations. All models operated on the identical dataset splits, processed the same 64×2048 spatial resolution, and executed the identical 5→5 temporal prediction task. No baseline method was provided with privileged temporal information or disparate sensor setups.

In terms of evaluation metrics, we systematically compare the Chamfer Distance and Bulldozer Distance loss values of TSMNet(Du et al., 2024), TCNet, MoNet, and MoCNet. Notably, the Bulldozer Distance is more sensitive to the local details and density distribution of point clouds(Liu et al., 2020), but due to its high computational complexity, this metric is calculated on 16,384 randomly sampled points in this experiment. As shown in Tab. II, when the low-frequency channel sharing rate of MoCNet is 0.125 and 0, its average Bulldozer Distance and Chamfer Distance are significantly lower than those of TCNet, MoNet(LSTM), TSMNet, and MoNet(GRU). This indicates that the point clouds predicted by the proposed method are closer to the real data in terms of spatial distribution.

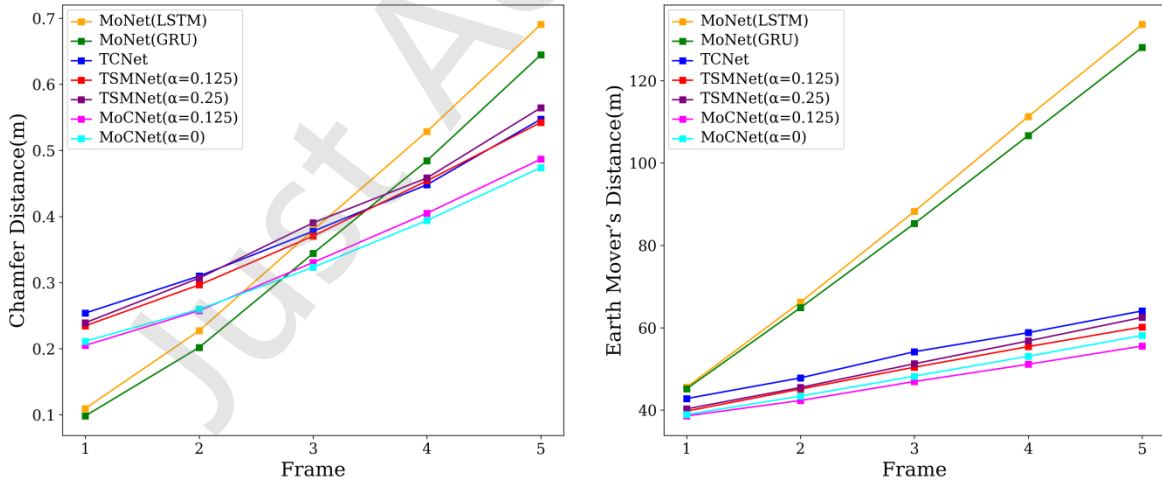


Fig. 7: Comparison of chamfer distance and earth mover distance for different methods on the test set from the fifth frame to the fifth frame

Further analysis combined with Fig. 7 reveals the internal mechanisms behind the performance degradation of baseline methods in long-term predictions (≥ 3 seconds). Traditional autoregressive architectures, such as MoNet(LSTM) and MoNet(GRU), rely heavily on standard recurrent units. During continuous sequence generation, minor spatial deviations produced in early frames are fed back into the network, compounding recursively within the hidden states. This inherent error accumulation leads to significant structural distortions over time. Similarly, 3D convolution-based methods like TCNet and TSMNet lack explicit motion modeling, causing the predicted shapes

to blur or deform as the temporal horizon extends. In contrast, MoCNet maintains the lowest levels of both Chamfer Distance and Earth Mover's Distance (EMD) in long-term forecasting. Most notably, at the 5-second prediction horizon, its EMD is reduced by 13% compared to the suboptimal method.

TABLE II: Comparison of EMD (m) and CD (m) for different methods on the test set.

Method	L_{CD}	L_{EMD}
--------	----------	-----------

MoNet	0.554	86.003 (GRU)
S2Net (Weng et al., 2022)	0.357	120.391
PCPNet (Luo et al., 2023)	0.422	—
TSMNet	0.392 ($\alpha = 0.25$)	51.266 ($\alpha = 0.25$)
	0.380 ($\alpha = 0.125$)	50.171 ($\alpha = 0.125$)
Const.vel Baseline	0.433	—
Ray Tracing Baseline	0.421	—
TCNet	0.387	53.535
Ours	0.337 ($\alpha = 0.125$)	46.909 ($\alpha = 0.125$)
	0.332 ($\alpha = 0$)	48.353 ($\alpha = 0$)

This substantial advantage is fundamentally driven by the internal mechanism of our motion-content dual-encoding architecture. Unlike standard RNNs or 3D convolutions that implicitly learn temporal features, MoCNet's motion encoder explicitly computes coordinate and

correlation differentials between adjacent frames. This explicit temporal differencing acts as a strict geometric constraint that anchors the predicted trajectories. By tracking the exact magnitude and direction of spatial changes (motion features), MoCNet effectively prevents the recursive drift and spatial deformation, thereby drastically mitigating the error accumulation issue and ensuring high structural fidelity in long-term predictions.

To comprehensively assess the performance of different point cloud prediction methods, this paper evaluates the Chamfer distance and the Bulldozer distance from the first to the fifth frame on the test set. The results, presented in Fig. 8, demonstrate that, except for TSMNet, TCNet, and MoCNet, the Chamfer and Bulldozer distances of other methods increase rapidly as the prediction time extends.

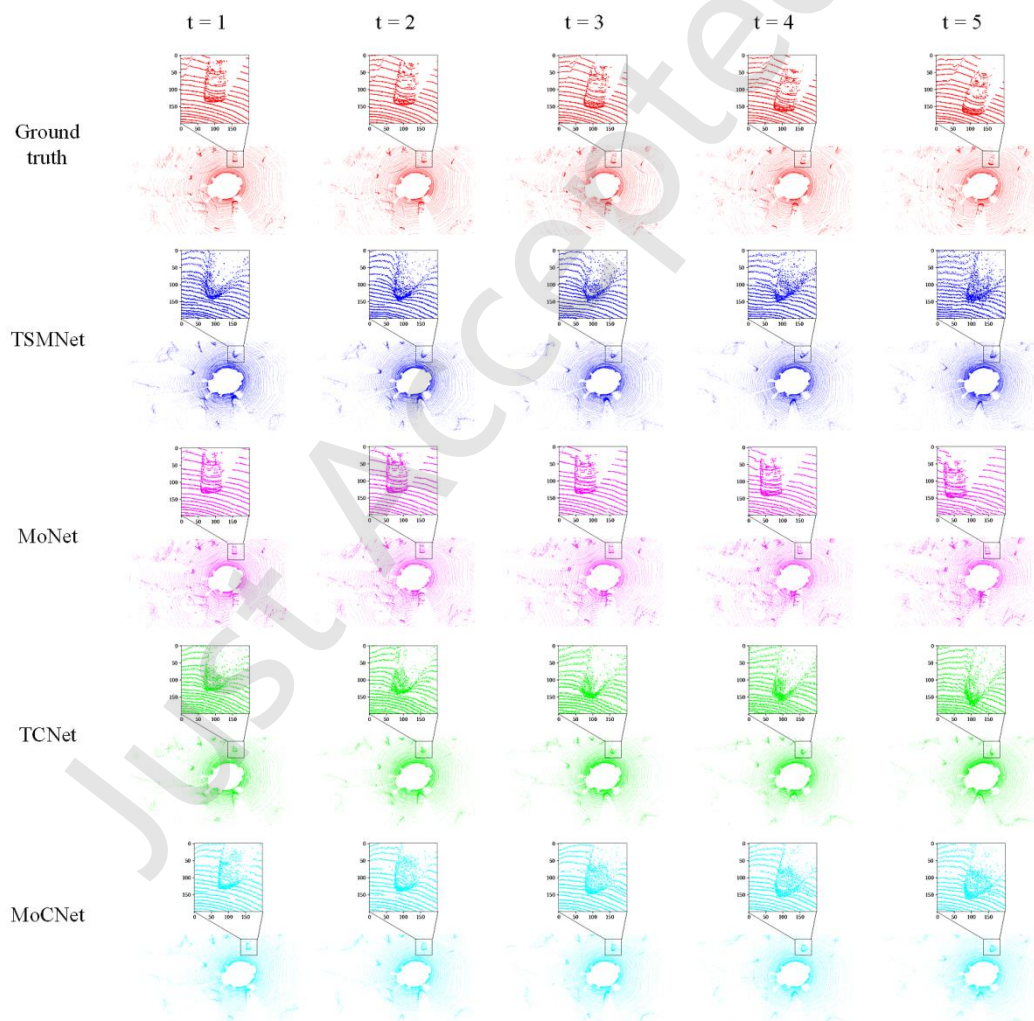


Fig. 8: Visualization of predicted point clouds as well as real point clouds on KITTI Odometry dataset by different methods

Notably, MoCNet exhibits a slower growth rate in both metrics, indicating superior performance in long-term point cloud prediction. Furthermore, to provide a more intuitive comparison, Fig.8 displays the predicted point clouds from different methods on the KITTI Odometry dataset, specifically during an autorickshaw's turning scenario, where the LiDAR coordinate system rotates. This scenario effectively highlights the performance of point cloud prediction algorithms. The figure shows five future frames from $t=1$ to $t=5$, with a zoomed-in view of the vehicle point cloud. From the zoomed-in region, it can be observed that TSMNet predicts the point cloud with a location and orientation similar to the ground truth but with significant shape discrepancies. In contrast, MoCNet produces a point cloud with a shape close to the real point cloud but deviates in location and orientation. TCNet's prediction is similar to

TSMNet in shape but slightly inferior in location and orientation. Among all methods, MoCNet demonstrates the closest resemblance to the real point cloud in terms of shape, orientation, and location, highlighting its superior performance.

4.3 Model generalization capability

In order to verify the generalization ability of the MoC-Net model on other datasets, in this paper, we will use the model fine-tuned on the KITTI Odometry dataset to test the generalization ability of the MoCNet model on the three datasets of Apollo-DaoxiangLake, namely, DaoxiangLake-1130, DaoxiangLake-0924, DaoxiangLake-1225 which are tested on the three datasets from 46146 to 58577 frames, 60306 to 73241 frames, and 40924 to 49504 frames, and the test results are shown as green bars in Fig. 9.

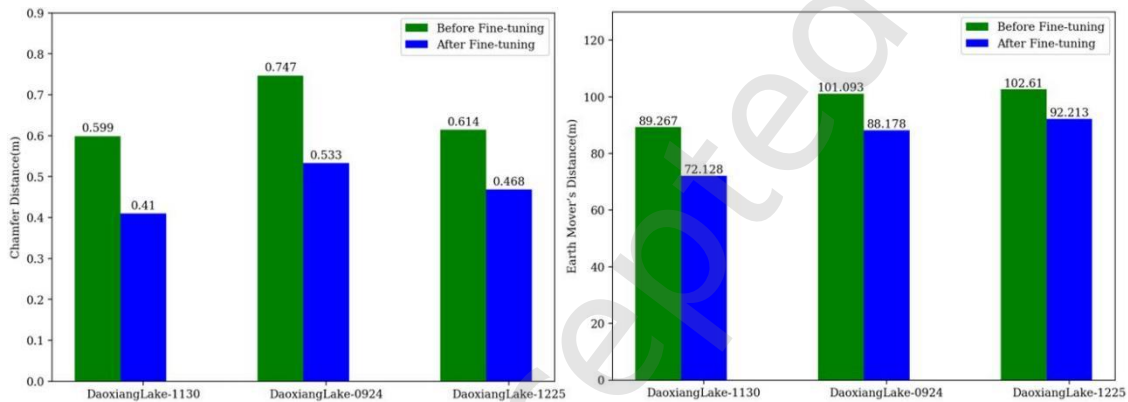


Fig. 9: Results of MoCNet model tested on Apollo-DaoxiangLake dataset

1) Zero-Shot Transfer Test: Without fine-tuning, the model achieved average Chamfer distances of 0.599m, 0.747m, and 0.614 m on the test frame segments (46146- 58577, 60306-73241, and 40924-49504) of the three sub-sets, respectively. The average Bulldozer distances were 89.267, 101.093, and 102.61 (represented by the green bars in Fig. 9). While the model demonstrates strong baseline adaptability, the observed performance discrepancy in the Zero-Shot Transfer-specifically the increase in average Chamfer distance compared to the KITTI dataset(German scenes)-requires deeper analysis. This decline is fundamentally driven by the pronounced domain gap between Chinese and German road scenarios. First, Chinese traffic environments (such as those in the Apollo-DaoxiangLake dataset) exhibit a significantly higher density of mixed traffic, characterized by a massive presence of two-wheeled vehicles (e.g., electric scooters, bicycles) and pedestrians. These targets are not only physically smaller, yielding sparser point cloud reflections, but they also exhibit highly non-linear, flexible, and unpredictable motion trajectories compared to the relatively structured and lane-adhering vehicle movements dominant in the KITTI dataset.

Consequently, under zero-shot conditions, the temporal-dynamics motion encoding stream encounters challenges in extracting stable temporal differential features for these erratic objects. Second, discrepancies in sensor configurations-such as variations in LiDAR mounting height, scanning frequency, and vertical angular resolution-alter the intrinsic spatial distribution, ground plane inclination, and

dynamic occlusion patterns of the point clouds. These combined factors inevitably lead to higher geometric prediction errors before the model adapts to the new domain via fine-tuning.

2) Single-Round Fine-Tuning Optimization: After a single round of parameter optimization on the training frame segments of the DaoxiangLake subsets (8063-38142, 8177-47600, and 6710-32356), the model performance improved significantly. The average Chamfer distances reduced to 0.41m, 0.533 m, and 0.468m, while the Bulldozer distances improved to 72.128, 88.178, and 92.213. This enhancement demonstrates the model's efficient domain adaptation capability provided by the self-supervised training mechanism, with performance surpassing TSMNet in all aspects (see Fig. 9 for detailed comparisons).

3) Real-Time Analysis: On a hardware platform equipped with an i7-12700 CPU and a GeForce RTX 3080 GPU (10GB), MoCNet exhibited outstanding computational efficiency.

- Low-Frequency Channel Priority Mode: Predicting 5 frames of point clouds took 20 ms (50 Hz), achieving a real-time performance 5 times higher than the LiDAR frame rate.
- Full-Bandwidth Optimization Mode: The inference time was further reduced to 15 ms (67 Hz), which is 6.7 times the typical 3D LiDAR sensor frame rate (10 Hz).

This performance advantage ensures seamless integration of the model into the real-time perception pipeline of autonomous driving systems, meeting the online prediction requirements in complex dynamic environments. To validate edge deployment feasibility, we

evaluated MoCNet on an NVIDIA Jetson AGX Xavier platform. Directly executing the native PyTorch model on this embedded SoC requires 2.58 seconds per forward pass, falling short of real-time requirements.

To optimize the memory allocation graph and execution efficiency, we implemented a structured deployment pipeline. The model was first exported to a simplified ONNX format (opset 17, supporting dynamic batching and temporal steps) and subsequently compiled into a highly optimized TensorRT engine. For a standard continuous input tensor $X \in \mathbb{R}^{1 \times 4 \times 5 \times 64 \times 2048}$, this TensorRT acceleration successfully reduced the inference latency from 2.58 seconds down to 0.211 seconds. Although this hardware-specific compilation incurs a minor trade-off in precision matching, it rigorously validates the feasibility of deploying complex spatio-temporal attention networks on resource-constrained automotive SoCs by drastically minimizing the memory footprint.

4.4. Ablation experiments

1) Ablation Study of MoCNet: To systematically evaluate the contribution of each architectural component in MoCNet, this experiment adopts a progressive addition strategy. We sequentially integrate the Spatial-Context Content Encoding Stream, the Temporal-Dynamics Motion Encoding Stream, the OctSTA decoder, and the OctConv mechanism into the TCNet baseline. The performance impact of each component is quantified by the reduction in Chamfer Distance loss on the validation set (Fig. 10).

2) Dual-Stream Encoding Enhancement: The Spatial-Context Content Encoding Stream and the Temporal-Dynamics Motion Encoding Stream are introduced into TCNet. These streams enhance the model's ability to extract intra-frame geometric correlations and inter-frame temporal dynamics from the input point cloud,

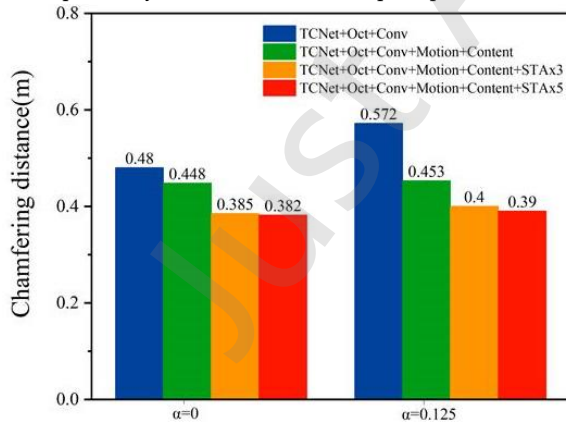


Fig.10: Effect of different components on model performance

Comprehensive empirical evaluations demonstrate that $\alpha=0.125$ establishes the optimal Pareto frontier. At this specific ratio, the network securely compresses redundant global background information into the low-frequency channels. Simultaneously, it maintains sufficient high-frequency bandwidth (87.5% of channels) to preserve structural fidelity. Increasing α beyond 0.125 results in a pronounced degradation in Chamfer Distance due to excessive spatial blurring. This rigorously validates $\alpha=0.125$ as the optimal architectural configuration. The concrete efficiency gain is explicitly demonstrated in the computational volume

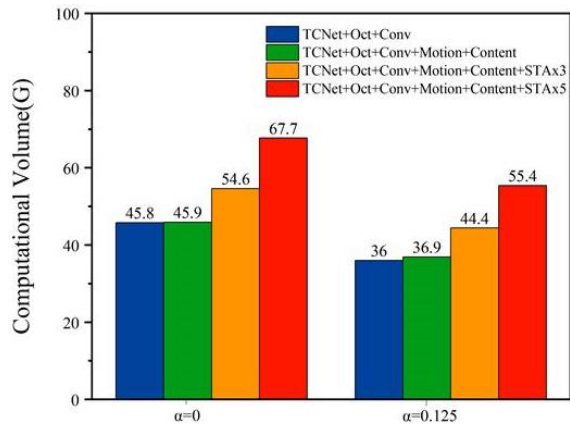
respectively. Experimental results show that integrating this dual-stream architecture reduces the Chamfer Distance loss by 20%. This substantial reduction validates the effectiveness of the proposed spatiotemporal decoupling and fusion paradigm.

3) Large-Kernel Attention Decoding: By inserting the OctSTA decoder, which utilizes large-kernel convolutions, spatial-temporal joint attention modeling is achieved. This block further reduces the Chamfer Distance loss by 9%. This improvement demonstrates that expanding the effective receptive field can successfully capture long-range dependencies and suppress irrelevant noise interference.

4) Octave Convolution Optimization:

We adopt a high-low frequency separated OctConv mechanism to replace traditional 3D convolutions. Consequently, the spatial resolution of the low-frequency pathway is reduced to half that of the high-frequency pathway ($1/2 \times 1/2$). This design establishes a cross-frequency interaction mechanism that effectively optimizes the computational graph.

The selection of the high-low frequency channel ratio, specifically $\alpha=0.125$, is fundamentally driven by the trade-off between computational efficiency and fine-grained geometric retention. The parameter α dictates the proportion of feature channels allocated to the low-frequency pathway. A larger α (e.g., $\alpha=0.25$ or 0.5) more aggressively reduces floating-point operations (FLOPs). However, it severely risks over-compressing the feature space. This over-compression leads to the loss of crucial high-frequency spatial details, such as the sharp boundaries of small moving objects (e.g., pedestrians). Conversely, setting $\alpha=0$ reverts the network to standard full-resolution 3D convolutions, thereby maximizing the computational burden.



analysis (Fig. 10). Here, we compare the full-bandwidth mode ($\alpha=0$, functionally equivalent to standard 3D convolutions) with the optimal OctConv mode ($\alpha=0.125$). As illustrated, introducing the OctConv mechanism at $\alpha=0.125$ significantly reduces the model's total computational load from 67.7G to 55.4G. Consequently, the model achieves an outstanding inference time of 15 ms to 20 ms (50 Hz to 67 Hz). This represents a processing speed 5 to 6.7 times faster than the standard 10 Hz frame rate of typical 3D LiDAR sensors. This explicitly proves the mechanism's capacity to drastically reduce computational overhead

while maintaining strong feature representation capabilities.

The above results indicate that the dual-stream encoding paradigm and the OctSTA decoder collectively drives the core performance improvements. Meanwhile, the OctConv mechanism provides a highly effective precision-efficiency trade-off for real-time autonomous driving applications.

5 Limitations and failure cases analysis

While MoCNet demonstrates state-of-the-art performance across standard evaluation metrics, a critical analysis of its failure cases reveals specific boundary limitations inherent to the current architecture. First, the model exhibits localized performance degradation when predicting highly dynamic, sparse objects at long distances (e.g., fast-moving pedestrians or cyclists beyond 50 meters). Due to the limited number of reflected LiDAR points occupying the range image pixels, the temporal-dynamics motion encoding stream occasionally struggles to establish reliable cross-frame temporal correlations. This deficiency can lead to vanishing objects or blurred geometric boundaries in the forecasted frames.

Second, severe dynamic occlusions pose a significant challenge. When large vehicles (such as buses or trucks) completely obscure other traffic participants for extended periods, the 2D range image projection fundamentally loses the occluded 3D structural continuity. Because MoCNet's self-supervised signals rely strictly on visible spatial topology, it cannot accurately hallucinate the trajectories of fully occluded objects. Consequently, this results in transient spikes in Chamfer Distance during such scenarios.

Finally, while the OctSTA block effectively suppresses error accumulation up to the 5-second horizon, extreme ultra-long-term predictions (e.g., >8 seconds) in dense traffic still inevitably suffer from morphological distortions. This phenomenon occurs because the autoregressive temporal differencing, although effective, lacks the explicit physical kinematic constraints (e.g., rigid body dynamics, friction, and mass) necessary to perfectly anchor objects over extended temporal windows. Acknowledging these limitations paves the way for future integrations of Physics-Informed Neural Networks (PINNs) and multi-sensor fusion to further enhance architectural robustness.

6 Conclusion

This study proposes a highly efficient, self-supervised point cloud prediction framework, MoCNet, to address the critical limitations of conventional supervised approaches that rely on extensive labeled data and computationally expensive 3D convolutions. The core innovation of MoCNet lies in its dual-stream motion-content encoding architecture and its large-kernel attention mechanisms. By seamlessly coupling the Spatial-Context Content Encoding Stream—which captures intra-frame geometric correlations via Farthest Point Sampling (FPS) and k-NN with the Temporal-Dynamics Motion Encoding Stream that explicitly calculates inter-frame temporal differentials, the network establishes a robust, physics-based inductive bias. This dual-stream mechanism effectively resolves the inherent sparsity challenges and prevents the

recursive structural deformations commonly observed in dynamic point cloud sequences.

The integration of the OctSTA decoder utilizes large-receptive-field convolutions to implicitly emulate global attention mechanisms, successfully suppressing temporal error accumulation while maintaining a linear computational complexity. Coupled with the high-low frequency separated Octave Convolution (optimally set at the Pareto frontier of $\alpha=0.125$), MoCNet achieves a drastic reduction in computational volume. Empirical evaluations demonstrate that the model operates at an outstanding inference speed of 15 to 20 ms (50 to 67Hz)—significantly surpassing the standard 10Hz frame rate of typical 3D LiDAR sensors without sacrificing geometric fidelity. Extensive experiments on the KITTI Odometry and Apollo-DoxiangLake datasets confirm that MoCNet substantially lowers both Chamfer Distance and Earth Mover's Distance, particularly excelling in challenging 5-second long-term predictions. The model also exhibits promising zero-shot generalization capabilities across diverse, multi-agent traffic domains.

Despite these significant advancements, the current framework exhibits specific boundary limitations, such as morphological distortions in ultra-long-term predictions (e.g., >8 seconds) and trajectory estimation failures under severe dynamic occlusions. Since the current temporal encoding relies on a fixed sampling interval, future work will also investigate time-aware temporal encoding by incorporating the actual time interval Δt into the frame-difference representation, thereby improving robustness to variable sampling rates and frame drops. To bridge the gap toward the subdecimeter precision required for mission-critical autonomous navigation, future research will prioritize the integration of PINNs to embed explicit rigid-body kinematic constraints. Additionally, exploring multi-sensor fusion (e.g., LiDAR-camera synergy) will be crucial to overcoming occlusion-induced structural losses and further enhancing the robustness of spatio-temporal predictions in complex, real-world environments.

Replication and data sharing

The source data within this research can be made accessible upon request via email to the corresponding author.

Acknowledgements

This work was supported in part by the National Natural Science Foundation of China (Grant No. 52572399), and in part by Innovation Fund of Research Institute for Innovative Development of Transportation and Energy Integration (Changsha University of Science & Technology)(JNRH11) and in part by Hunan Provincial Innovative Research Group Project of China (Grant No. 2025JJ10006).

Declaration of competing interest

The authors have no competing interests to declare that are relevant to the content of this article.

References

- Abbasifard, M. R., Ghahremani, B., and Naderi, H., 2014. A survey on nearest neighbor search methods. *International Journal of Computer Applications*, 95(25), 39-52.
- Cao, C., Preda, M., Zaharia, T., 2019. 3d point cloud compression: A survey. In Proceedings of the 24th International Conference on 3D Web Technology, 1 - 9.
- Chen, Y., Fan, H., Xu, B., Yan, Z., Kalantidis, Y., Rohrbach, M., et al., 2019. Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution. In Proceedings of the IEEE/CVF international conference on computer vision, 3435 - 3444.
- Cui, Y., Chen, R., Chu, W., Chen, L., Tian, D., Li, Y., et al., 2021. Deep learning for image and point cloud fusion in autonomous driving: A review. *IEEE Transactions on Intelligent Transportation Systems*, 23(2), 722 - 739.
- Deng, D., Zakhori, A., 2020. Temporal lidar frame prediction for autonomous driving. In 2020 International Conference on 3D Vision, 829 - 837.
- Ding, J., Zhang, J., Ye, L., Wu, C., 2024. Kalman-based scene flow estimation for point cloud densification and 3d object detection in dynamic scenes. *Sensors*, 24(3), 916.
- Du, R., Feng, R., Gao, K., Zhang, J., Liu, L., 2024. Self-supervised point cloud prediction for autonomous driving. *IEEE Transactions on Intelligent Transportation Systems*. 25(11), 17452- 17467.
- Fan, H., Yang, Y., 2019. Pointtrnn: Point recurrent neural network for moving point cloud processing. arXiv preprint arXiv:1910.08287.
- Fei, B., Yang, W., Liu, L., Luo, T., Zhang, R., Li, Y., et al., 2023. Self-supervised learning for pre-training 3d point clouds: A survey. arXiv preprint arXiv:2305.04691.
- Guo, M. H., Xu, T. X., Liu, J. J., Liu, Z. N., Jiang, P. T., Mu, T. J., et al., 2022. Attention mechanisms in computer vision: A survey. *Computational visual media*, 8(3), 331 - 368.
- Khurana, T., Hu, P., Held, D., Ramanan, D., 2023. Point cloud forecasting as a proxy for 4d occupancy forecasting. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 1116 - 1124.
- Li, Z., Liu, F., Yang, W., Peng, S., Zhou, J., 2021. A survey of convolutional neural networks: analysis, applications, and prospects. *IEEE transactions on neural networks and learning systems*, 33(12), 6999 - 7019.
- Lin, F., Yue, Y., Hou, S., Yu, X., Xu, Y., Yamada, K. D., et al., 2023. Hyperbolic chamfer distance for point cloud completion. In Proceedings of the IEEE/CVF international conference on computer vision, 14595 - 14606.
- Liu, J. M., Han, J. R., Liu, L. H., Aviles, A., Jiang, C. K., Liu, Z., 2025. Mamba4d: Efficient 4d point cloud video understanding with disentangled spatial-temporal state space models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 17626 - 17636.
- Liu, M., Sheng, L., Yang, S., Shao, J., Hu, S. M., 2020. Morphing and sampling network for dense point cloud completion. In Proceedings of the AAAI conference on artificial intelligence, volume 34, 11596 - 11603.
- Liu, X., Qi, C. R., Guibas, L. J., 2019. FlowNet3D: Learning scene flow in 3d point clouds. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 529 - 537.
- Liu, X., Yan, M., Bohg, J., 2019. MeteorNet: Deep learning on dynamic 3d point cloud sequences. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 9246 - 9255.
- Lu, F., Chen, G., Li, Z., Zhang, L., Liu, Y., Qu, S., et al., 2021. Monet: Motion-based point cloud prediction network. *IEEE Transactions on Intelligent Transportation Systems*, 23(8), 13794 - 13804.
- Luhmann, T., Robson, S., Kyle, S., Boehm, J., 2023. Close-range photogrammetry and 3D imaging. Walter de Gruyter GmbH & Co KG, 102-120.
- Luo, Z., Ma, J., Zhou, Z., Xiong, G., 2023. Pcpnet: An efficient and semantic-enhanced transformer network for point cloud prediction. *IEEE Robotics and Automation Letters*, 8(7), 4267 - 4274.
- Mahjourian, R., Kim, J., Chai, Y., Tan, M., Sapp, B., Anguelov, D., 2022. Occupancy flow fields for motion forecasting in autonomous driving. *IEEE Robotics and Automation Letters*, 7(2), 5639 - 5646.
- Rao, Y., Lu, J., Zhou, J., 2019. Spherical fractal convolutional neural networks for point cloud recognition. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, 452 - 460.
- Qi, C. R., Li, Y., Su, H., Guibas, L., 2023. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 1-10.
- Sheng, X., Shen, Z., Xiao, G., Wang, L., Guo, Y., Fan, H., 2023. Point contrastive prediction with semantic clustering for self-supervised learning on point cloud videos. In Proceedings of the IEEE/CVF International Conference on Computer Vision, 16515 - 16524.
- Sun, X., Wang, S., Wang, M., Wang, Z., Liu, M., 2020. A novel coding architecture for lidar point cloud sequence. *IEEE Robotics and Automation Letters*, 5(4), 5637 - 5644.
- Wang, X., Wu, Z., Jin, B., Lin, M., Zou, F., Liao, L., 2024. Mdstf: a multi-dimensional spatio-temporal feature fusion trajectory prediction model for autonomous driving. *Complex & Intelligent Systems*, 1 - 19.
- Weng, X., Nan, J., Lee, K. H., McAllister, R., Gaidon, A., Rhinehart, N., et al., 2022. S2net: Stochastic sequential pointcloud forecasting. In European Conference on Computer Vision, Springer, 549 - 564.
- Wu, Q., Li, X., Wang, K., Bilal, H., 2023. Regional feature fusion for on-road detection of objects using camera and 3d-lidar in high-speed autonomous vehicles. *Soft Computing*, 27(23), 18195 - 18213.
- Yuan, C., Chen, G., Yi, L., Gao, Y., 2025. Self-supervised monocular 4d scene reconstruction for egocentric videos. In Proceedings of the

Zhong, F., Bi, X., Zhang, Y., Zhang, W., Wang, Y., 2023. Rspt: reconstruct surroundings and predict trajectory for generalizable active object tracking. In Proceedings of the AAAI Conference on Artificial Intelligence, 37, 3705 – 3714.

Zhou, Y., Wan, G., Hou, S., Yu, L., Wang, G., Rui, X., et al. 2020. Da4ad: End-to-end deep attention-based visual localization for autonomous driving. In Computer Vision-ECCV 2020: 16th European Conference, Glasgow, UK, 271-289.



Kai Gao was born in 1985 in Baoding, Hebei province, China. He received the B.S. degree from Central South University, Changsha, China in 2008, the Ph.D. degree from Central South University, Changsha, China in 2014. He joined the Changsha University of Science & Technology in 2015. His research interests include connected and automated vehicles, intelligent transportation systems theory and applications.



Yuxiang Zhou is currently pursuing the M.S. degree with the Changsha University of Science and Technology. His research interests include point cloud prediction and deep learning.



Pingping Dong received her B.S., M.S. and Ph.D. degree from the School of Information Science and Engineering at Central South University, P.R. China. Currently she is an associate professor in the college of information science and engineering, Hunan Normal University, Changsha, P.R. China. Her research interests include Intelligent Transportation System.



Lin Hu (Member, IEEE) received the Ph.D. degree in mechanical engineering from Hunan University,

Changsha, China, in 2008. He is currently a Professor with the School of Automotive and Mechanical Engineering, Changsha University of Science and Technology. His research interests include intelligent vehicles, vehicle dynamics, and traffic safety. He received the first prize for scientific and technological progress in Hunan Province in 2021 and the National Outstanding Youth Fund in 2023.



Rongying Feng received the B.E. degree in vehicle engineering from Changsha University of Science and Technology, Changsha, China, in 2021, where he is currently pursuing the M.E. degree in vehicle engineering. His research interests include 3D computer vision and deep learning.



Jinlai Zhang received the B.S. degree from Changsha University of Science and Technology, China, in 2017 and the Ph.D. degree from Guangxi University, China, in 2023. He is now a lecturer with the College of Automotive and Mechanical Engineering, Changsha University of Science and Technology. His research interests include 3D deep learning and time series forecasting.



Kaihan Zhang is currently a Ph.D. candidate at KAIST, South Korea. His research interests include transport planning, behavior modeling, and intelligent transportation systems.



Inhi Kim received his Ph.D. in Civil Engineering from the University of Queensland, Australia, in 2014. He is currently an Associate Professor at KAIST. His expertise includes transport planning, metaverse, connected vehicles, and driving simulators.