Self-Supervised Point Cloud Prediction for Autonomous Driving

Ronghua Du, Rongying Feng, Kai Gao, Jinlai Zhang, Linhong Liu

***Abstract*—Pose prediction and trajectory forecasting represent pivotal tasks in the realm of autonomous driving, crucially enhancing the planning and decision-making capabilities of self-driving vehicles. However, a prevailing challenge is that many existing algorithms for these tasks necessitate supervised training, demanding substantial human effort and computational resources. To alleviate this resource-intensive burden, this paper introduces an innovative method for predicting future object poses and trajectories in a 3D space, obviating the requirement for manual annotations. The enhanced self-supervised 3D point cloud prediction algorithm proposed in this study incorporates an 3D Action Attention module, augmenting TCNet's proficiency in extracting vital spatiotemporal and motion information from continuous point cloud range images. Additionally, 3D Octave Convolution is harnessed to mitigate the computational overhead introduced by the 3D Action Attention module, consequently accelerating the model's inference speed. This advanced self-supervised 3D point cloud prediction algorithm is denoted as TSMNet. TSMNet outperforms the baseline TCNet and several SOTA 3D point cloud prediction models when evaluated on the KITTI Odometry dataset. Moreover, it showcases robust generalization capabilities in unfamiliar environments. Notably, TSMNet can predict future point cloud data for five frames in a mere 33 milliseconds, surpassing the frame rate of typical LiDAR sensors, which typically operate at 10Hz. Furthermore, when integrated with a point cloud clustering and tracking algorithm, the improved self-supervised 3D point cloud prediction algorithm facilitates the extraction of object poses and trajectories. The performance metrics of the point cloud clustering and tracking algorithm attain remarkable levels of accuracy, with a Multiple Object Tracking Accuracy (MOTA) of 86.12% and a Multiple Object Tracking Precision (MOTP) of 91.01% on the KITTI dataset.**

***Index Terms*—Autonomous Driving, Prediction, Point Cloud, Pose, Trajectory.**

I. Introduction

CONTINUOUS 3D point clouds collected by sensors like Light Detection and Ranging (LiDAR), RGB-D cameras, and stereo cameras have found diverse applications in various aspects of autonomous driving. These applications encompass mapping [1], 3D object detection [2], 3D multi-object tracking [3], semantic segmentation [4], [5], surveying [6], and pose prediction and trajectory prediction [7]. Among these tasks, utilizing 3D point cloud data acquired from sensors for pose prediction and trajectory forecasting is of significant importance in achieving the safety, efficiency, and intelligence of autonomous driving vehicles. Pose prediction accurately anticipates changes in vehicle orientation, tilt, and position. By continuously monitoring and predicting vehicle poses in real-time, the system can make timely adjustments to ensure the vehicle operates in a safe and stable state. Trajectory forecasting predicts the future paths and trajectories of vehicles, which is crucial for planning vehicle routes, avoiding collisions, and optimizing traffic flow. Through accurate prediction of vehicle trajectories, the system can proactively plan and make decisions in advance, avoiding traffic accidents and congestion, thereby enhancing overall driving efficiency and safety [8], [9]. However, the majority of existing algorithms designed for pose and trajectory prediction necessitate prior object detection and tracking within the 3D point cloud scene [10]–[12]. Subsequently, the results of these detection and tracking processes are input into predictive models. These algorithms require extensive supervised training of detection and tracking models, as well as prediction models. The training of prediction models necessitates continuous annotation information of targets over a period of time, which requires a significant amount of manpower and resources to obtain in 3D space. Therefore, the method of first detection and tracking, followed by prediction, cannot enable real-time updates of the models to adapt more quickly to unknown environments. To reduce human and material resource consumption and achieve real-time model updates, this paper proposes a method utilizing an improved self-supervised [13]–[15] point cloud prediction algorithm and point cloud clustering and tracking algorithm to predict future object poses and trajectories in 3D space without the need for annotated information.

The prediction of high-dimensional and sparse 3D point clouds poses a challenging problem. Incorporating attention mechanisms into 3D point cloud prediction models can assign different attention weights to important and unimportant information, thus improving the effectiveness of point cloud prediction. However, previous 3D point cloud prediction methods [16], [17] have not explored the integration of attention mechanisms into point cloud prediction models. Addressing this gap, this paper proposes the TSMNet neural network based on attention mechanisms. In this paper, we first project 3D point clouds onto 2D range images, transforming unordered 3D point clouds into ordered 2D range images suitable for convolutional operations. This approach reduces the complexity of point cloud processing and accelerates model inference speed. Subsequently, we input the range images of the previous

This work was supported in part by the Natural Science Foundation of Hunan under Grant 2024J15023 and the Excellent Youth Foundation of Hunan Provincial Education Department under Grant 21B0286. *(Corresponding author: Jinlai Zhang and Kai Gao.)*

Ronghua Du, Rongying Feng, Kai Gao, Jinlai Zhang and Linhong Liu are with the College of Automotive and Mechanical Engineering, Changsha University of Science and Technology, Changsha 410114, China (e-mail: csdrh@csust.edu.cn; 15015695620@163.com; kai_g@csust.edu.cn; jinlai.zhang@csust.edu.cn; llh_4053@163.com).

P frames into the encoder-decoder neural network TSMNet, which integrates TCNet [16], 3D Action Attention module [18], and 3D Octave Convolution module [19]. Integrating the 3D Action Attention module into TCNet enables the encoder-decoder neural network TSMNet to selectively focus on important spatiotemporal and motion information while suppressing irrelevant information, thus enhancing point cloud prediction. The addition of the 3D Octave Convolution module reduces the computational overhead introduced by the 3D Action Attention module, speeding up model inference. In the ensuing stages, the encoder-decoder neural network TSMNet generates a mask image that discriminates between valid and invalid points within the range image. It subsequently produces future *F* frame range images containing exclusively valid points. These future range images are then back-projected to yield the forthcoming 3D point clouds. The resulting point cloud is further processed through a point cloud clustering and tracking algorithm, which facilitates the derivation of the future poses and trajectories of objects.

To evaluate the effectiveness of the 3D point cloud prediction model, extensive experiments were conducted using the KITTI Odometry dataset. Compared to previous SOTA models, our model achieved superior results while requiring less inference time. Additionally, the generalization capability of the model was validated using the Apollo-DaoxiangLake dataset. The experiments demonstrated the model's strong generalization ability, as it adapted well to unknown environments after a single round of fine-tuning.

The principal contributions of this paper can be summarized as follows:

- Introducing a novel encoder-decoder neural network, TSMNet, capable of predicting future point clouds (*F* frames) based on past point clouds (*P* frames), surpassing the performance of previous point cloud prediction methods, with faster inference speeds compared to common LiDAR frame rates (10 Hz), and it can be effectively generalized to unknown environments.
- Using the point clouds predicted by TSMNet in downstream tasks, we obtain the future pose and trajectory of objects in 3D space, achieving a pose prediction and trajectory prediction distinct from traditional prediction methods.
- Utilizing a unified approach that combines a point cloud prediction model and a point cloud clustering and tracking algorithm to cluster and track the predicted future point clouds. This process results in the extraction of object poses and future trajectories in 3D space. The predicted point clouds encompass the entire scene, thereby minimizing potential instances of target omission.

In the structure of the paper, the introduction of relevant prior work is presented in Section II. In Section III, a detailed description of the proposed method, the associated loss functions, and the relevant performance metrics of the experiments are provided. Section IV conducts an in-depth analysis of the experimental results. Finally, in Section V, the paper concludes by summarizing its contributions and identifying potential limitations.

II. Related Work

This section briefly reviews work related to point cloud prediction, including trajectory prediction, point cloud scene flow estimation, point cloud prediction, and point cloud object detection and tracking.

A. Trajectory Prediction

Trajectory prediction involves anticipating the future trajectories of moving objects like vehicles and pedestrians. Conventional trajectory prediction approaches often depend on kinematic models (e.g., CV(Constant Velocity), CA(Constant Acceleration), CTRV(Constant Turn Rate and Velocity), CTRA(Constant Turn Rate and Acceleration)) and the present state of the target to forecast trajectories within a limited time frame, typically suitable for short-term predictions not exceeding one second. If the prediction horizon is extended, assumptions solely based on kinematic models may no longer hold true. Recently, many deep learning-based trajectory prediction methods have been capable of predicting longer and more accurate trajectories. Among these, Chandra et al. [20] introduced a technique that combines deep learning and spectral graphs to predict trajectories of dynamic obstacles in urban areas. They assessed the closeness of dynamic obstacles by employing a dynamic geometric graph with interconnecting weights among the obstacles. Additionally, a two-stage graph LSTM was employed for trajectory prediction. The SCOUT model [21] integrated interactions among dynamic obstacles to ensure motion planning that is both safe and comfortable. This model incorporated Graph Neural Networks (GNNs) and attention mechanisms to predict socially consistent trajectories of dynamic obstacles under moderate traffic flow. DATF [22] introduced a method for synthesizing input signals from a multimodal world by establishing interactions among agents in the scene using attention mechanisms. This approach provides more accurate approximation techniques to capture the features of real trajectory distributions better. This approach enhances prediction accuracy and enables the acquisition of richer information from multiple data sources, resulting in more comprehensive and reliable predictions. However, the aforementioned methods are designed primarily for predicting dynamic objects and do not encompass the entire future scene. Moreover, they require labeled data for training.

B. Point Cloud Scene Flow Estimation

Point Cloud Scene Flow Estimation entails establishing dense correspondences between successive frames of point clouds, calculating motion vectors, and determining depth information for individual points. Wang et al. [23] introduced a scene flow estimation network that utilizes hierarchical attention learning. This network incorporates two separate attention mechanisms within each scene flow embedding. They designed a hierarchical attention flow refinement module to facilitate the layer-by-layer propagation and optimization of scene flow. SAFIT [24] introduced the concept of relationship reasoning, establishing connections between object-level and point-level relationships. This module captures relationship

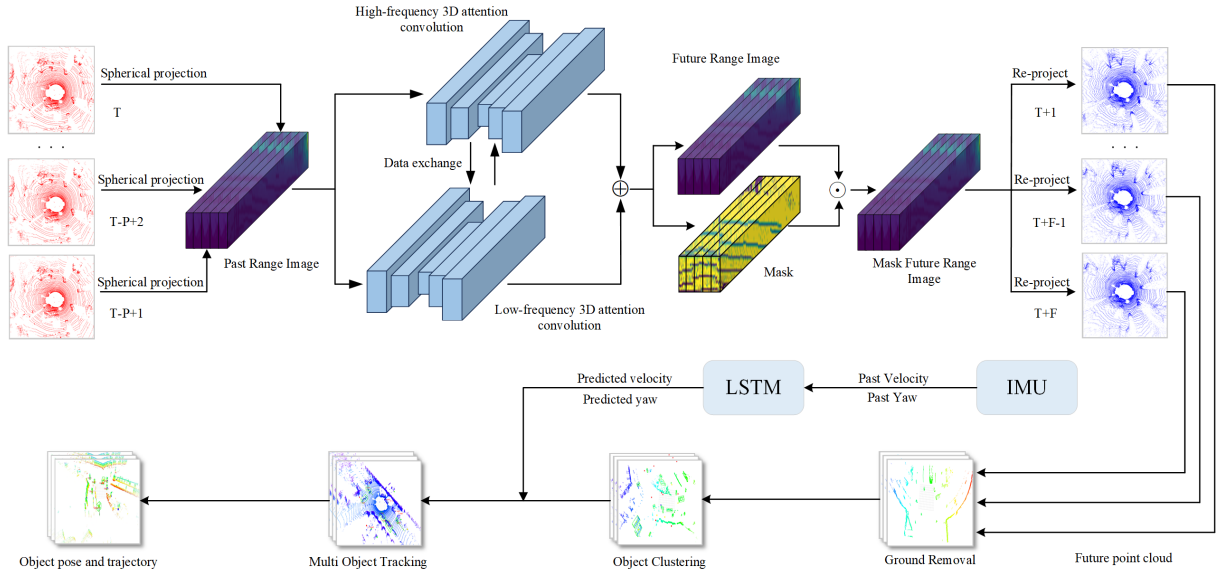

Fig. 1. Overall framework of our approach in this paper. At time T, the past P-frame point cloud is projected onto a range image, followed by concatenation in the temporal dimension. The concatenated range images are then processed through the Three-Dimensional Spatio-Temporal Attention Convolutional Neural Network (TSMNet), yielding predicted masks and range images. These are combined and reprojected to obtain future 3D point cloud predictions. Post-processing steps involve ground removal, clustering, and multi-object tracking, providing information about the future poses and 3D trajectories of objects. Additionally, incorporating the future velocity of the ego vehicle and its future yaw angle allows the determination of relative motion states between the ego vehicle and surrounding objects.

features among objects, enriching the feature combinations of 3D point clouds and ultimately improving scene flow estimation performance when combined with other features. RCP [25] disassembled scene flow estimation into two inter-dependent stages: the initial step fine-tunes the 3D flow at a point-wise level, and the subsequent step globally refines the 3D flow using recurrent networks. Within the point-wise optimization module, auxiliary flow vectors are calculated by connecting point features and position encoding. In the second optimization step, RCP utilizes GRU to update hidden states for estimating residual flow vectors. FLOT [26] redefined the scene flow estimation task as an optimal transport problem. It derived point features through multiple convolutional layers and evaluated transportation costs using cosine similarity. FlowStep3D [27] introduced a recursive structure that extends the scene flow estimation model through refinement operations. In FlowStep3D, initial flow vectors are estimated using a global correlation matrix, and the subsequent flow sequences are updated based on local correlations within gated recursive units. SCTN [28] introduced voxel-based convolutions to generate consistent flows in 3D space. It utilized a combination of sparse convolutions for feature extraction and Transformer modules for accurate scene flow prediction. However, a limitation of many existing point cloud scene flow estimation methods is their restricted coverage of the complete future scene.

C. Point Cloud Prediction

Point cloud prediction involves utilizing computer vision and deep learning techniques to forecast future point cloud data by analyzing historical data and scene information. Weng et al. [17] employed an encoder-decoder architecture to predict future point clouds and utilized the predicted point clouds for trajectory prediction. Lu et al. [29] applied a motion-based neural network, MoNet, which integrates motion features between two consecutive point clouds into the prediction pipeline. The introduction of motion features enables the model to more accurately capture changes in inter-frame motion information, leading to improved predictions of future motion. Benedikt Mersch et al. [16] utilized an encoder-decoder architecture using 3D convolutions to jointly capture spatial and temporal information of the scene and predict future 3D point clouds. Tarasha Khurana et al. [30] redefined the point cloud prediction problem as a task of predicting geometric occupancy and introduced a novel concept - future spatiotemporal 4D occupancy. Zou et al. [31] introduced a straightforward yet highly efficient Rigidly Point cloud Prediction Network (RPP-Net). The central element of this method is a versatile motion decoder. It creates motion masks and integrates both flow and rigid motion to generate a blended motion, all without necessitating extra annotations. These various methods of point cloud prediction have inspired the design of the approach presented in this paper. However, none of the aforementioned methods have investigated the impact of attention mechanisms on point cloud prediction models.

D. Point Cloud Object Detection and Tracking

Point Cloud Object Detection and Tracking refer to the identification of objects of interest within point clouds and the tracking of their movements over time. Yin et al. [32] introduced CenterPoint, a two-stage approach. In the initial stage, it employs a keypoint detector to pinpoint the object centers and estimate other attributes, such as 3D dimensions, orientation, and velocity. The subsequent stage refines these

estimations using additional object features, simplifying the final 3D object tracking to a greedy nearest point matching process. Felicia Ruppel et al. introduced TransMOT [33], a novel end-to-end trainable online point cloud data tracker and detector based on Transformers. This model leverages cross-attention mechanisms and self-attention mechanisms and is suitable for laser radar data in the field of autonomous driving, as well as other data types. This model incorporates cross-attention and self-attention mechanisms, making it suitable for various data types, including laser radar data in the field of autonomous driving. Chen et al.'s VoxelNeXt [34], a sparse convolution network, performs detection and tracking of 3D objects solely based on voxel features. This negates the necessity for sparse-to-dense conversion or subsequent NMS post-processing steps. However, point cloud object detection and tracking typically require annotations for training, and the annotation cost in 3D space is relatively high. Additionally, models trained on such data may suffer from instances of both false negatives and false positives.

III. Methodology

The primary objective of this paper is to accomplish self-supervised 3D point cloud prediction alongside multi-object detection and tracking. Given a sequential set of past point clouds, the overarching aim is to prognosticate forthcoming 3D point cloud sequences and concurrently perform clustering and tracking operations on these forecasted future 3D point clouds. The details of the method are shown in Algorithm 1.

As depicted in Fig. 1, the process commences with the projection of previous LiDAR point clouds onto 2D range images [16], [35]. Subsequently, these 2D range images are amalgamated into a 3D spatiotemporal tensor, which is subsequently fed into the encoder-decoder architecture. This architecture comprises TCNet, 3D Octave convolutions, and 3D Action attention modules (as elaborated in Section III-B), all designed to extract spatiotemporal information across range images at distinct time intervals. The resulting output constitutes the prediction of future range images, which are subsequently reversed into 3D point clouds. In the final stages, the reverse-projected 3D point clouds undergo clustering and tracking, a process detailed in Section III-C. The intricacies of the point cloud projection onto range images and the subsequent reversal process are expounded upon in Section III-A.

A. Point cloud projection into range images and subsequent back-projection

For a mechanical LiDAR sensor like the Velodyne HDL-64E, once it completes a full scanning revolution, the resulting 3D point cloud can be projected onto a hollow cylinder centered around the sensor using spherical mapping. This mapped information can be further unwrapped to generate a 2D image plane, commonly known as a "range image." The mathematical transformation of a 3D point in the point cloud, represented as $p = (x, y, z)$, into range image coordinates (u, v) is precisely defined by the following formula:

$$(u,v)^{\top} = \left(\tfrac{1}{2}\left[1-\arctan(y,x)\pi^{-1}\right]W, \left[1-\left(\arcsin\left(zr^{-1}\right)+\mathrm{f}_{\mathrm{up}}\right)\mathrm{f}^{-1}\right]H\right)^{\top} \quad (1)$$

Algorithm 1 Overall framework of our method

Input: Historical point cloud obtained from LiDAR scans: P
Output: The future pose matrix of the object: S

pplist = $[P_0, \cdots, P_n]$;
for P_i in enumerate(pplist) **do**
  R_i^P = proj(P_i)
end for;
return Past range image list: rplist = $[R_0^P, \cdots, R_n^P]$;
M^P = concat rplist;
encparalist = $[E_0, \cdots, E_m]$;
for E_i in enumerate(encparalist) **do**
  3D OctConv $\leftarrow E_i$
  3D Action $\leftarrow E_i$
  if i == 0 **then**
    X_i^H, X_i^L = 3D OctConv(3D Action (M^P))
  else
    X_i^H, X_i^L = 3D OctConv(3D Action (X_i^H, X_i^L))
  end if;
  if $0 \leq i \leq m-1$ **then**
    clone X_i^H, X_i^L
  end if;
end for;
return Clone data list: clonelist = $[X_0^H, X_0^L, \cdots, X_{m-1}^H, X_{m-1}^L]$, Encoder output: X_m^H, X_m^L;
decparalist = $[D_0, \cdots, D_n]$;
for D_i in enumerate(decparalist) **do**
  3D OctConv $\leftarrow D_i$
  ConvTranspose3d $\leftarrow D_i$
  3D Action $\leftarrow D_i$
  if i == 0 **then**
    X_i^H, X_i^L = ConvTranspose3d (X_m^H, X_m^L))
    X_i^H, X_i^L = 3D OctConv(3D Action (X_i^H, X_i^L))
  else
    X_i^H, X_i^L = 3D OctConv(3D Action (concat (X_i^H, X_{m-i}^H),concat(X_i^L, X_{m-i}^L)))
    X_i^H, X_i^L = ConvTranspose3d (X_i^H, X_i^L)
    if $i == m$ **then**
      M^F = 3D OctConv(3D Action (X_i^H, X_i^L))
    else
      X_i^H, X_i^L = 3D OctConv(3D Action (X_i^H, X_i^L))
    end if;
  end if;
end for;
return M^F;
Separate M^F to obtain future range image list: rflist = $[R_0^F, \cdots, R_n^F]$;
for R_i in enumerate(rflist) **do**
  F_i = reproj(R_i)
end for;
return Future n-frame point cloud list: pflist = $[F_0, \cdots, F_n]$;
Clustering and tracking(pflist);
return Pose matrix list: pmlist = $[S_0, \cdots, S_n]$;
// S includes the x, y, z coordinates, yaw angle, length, width, height, and velocity of the tracked object

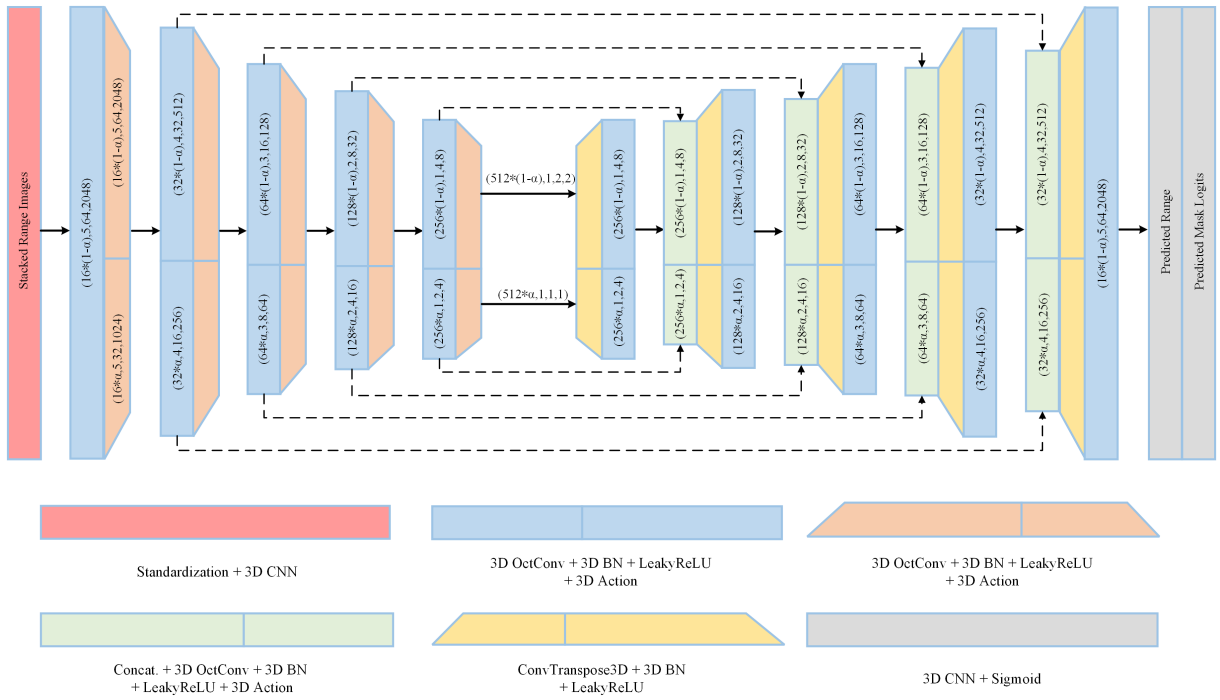

Fig. 2. The figure above illustrates the overall structure of TSMNet. In the diagram, solid lines represent the flow of information, while dashed lines indicate skip connections. Each colored box encloses details of a specific component that constitutes the network.

In the context of the range image, (H, W) denotes its height and width, and $f = f_{up} + |f_{down}|$ corresponds to the vertical field of view of the LiDAR. Within each pixel of the range image, the recorded value r indicates the Euclidean distance of the projected point, which is computed as $r = |\boldsymbol{p}|_2$. In cases where multiple points are mapped to the same pixel, the back-projection process retains the closest point. Furthermore, it is possible to reverse this process and convert 2D range images back into 3D point clouds. The formula for this back-projection is as follows:

$$(x, y, z)^\top = (r\cos(\theta)\cos(\gamma), r\cos(\theta)\sin(\gamma), r\sin(\theta))^\top \tag{2}$$

In this context, θ represents the pitch angle, which is determined by the arctan of the coordinates (y, x). Additionally, the yaw angle γ is computed as the arcsin of the ratio between z and r.

B. The proposed TSMNet

In this paper, the proposed method initially transforms 3D point clouds into 2D range images using spherical projection. Subsequently, it employs the encoder-decoder architecture depicted in Fig. 2 to extract spatiotemporal features from the past P-frame point cloud range images and generate future F-frame point cloud range images.

Compared to the 3D convolution-based encoder-decoder architecture TCNet, the incorporation of the 3D Action Attention module significantly enhances the encoder-decoder architecture's capability to capture point motion within the range images, thereby yielding more precise predictions of future range images. Moreover, the utilization of 3D Octave Convolution serves to mitigate the computational overhead introduced by the 3D Action Attention module, consequently expediting model inference. It is worth noting that this paper introduces modifications to both the Action Attention module [18] and the Octave Convolution module [19], as illustrated in Fig. 3 and Fig. 4, respectively.

The 3D Action Attention module consists of three sub-modules: Spatio-Temporal Excitation (STE) module, Channel Excitation (CE) module, and Motion Excitation (ME) module. The Spatio-Temporal Excitation (STE) module performs average pooling along the channel dimension of the input data $X \in \mathbb{R}^{N\times T\times C\times H\times W}$, yielding $X_S \in \mathbb{R}^{N\times 1\times T\times H\times W}$, followed by reshaping. Subsequently, it employs a $3 \times 3 \times 3$ 3D convolutional module K_S to obtain $X_S^* \in \mathbb{R}^{N\times 1\times T\times H\times W}$. This process can be represented by the following equation:

$$X_S^* = K_S * X_S \tag{3}$$

Next, X_S^* is transformed into $X_S^{**} \in \mathbb{R}^{N\times T\times 1\times H\times W}$ and processed with the Sigmoid function to obtain a spatiotemporal attention map $M_S \in \mathbb{R}^{N\times T\times 1\times H\times W}$. Subsequently, the resultant product of the spatiotemporal attention map and the initial input data is added to the initial data to emphasize significant spatiotemporal features. The output of the STE module, $Y_S \in \mathbb{R}^{N\times T\times C\times H\times W}$, can be represented as:

$$Y_S = X + X \odot M_S \tag{4}$$

The Channel Excitation module (CE) performs average pooling along the H and W dimensions of the input data $X \in \mathbb{R}^{N\times T\times C\times H\times W}$, yielding $X_C \in \mathbb{R}^{N\times T\times C\times 1\times 1}$. This can be represented mathematically as follows:

$$X_C = \frac{1}{H\times W}\sum_{i=1}^{H}\sum_{j=1}^{W} X\,[:,:,:,i,j] \tag{5}$$

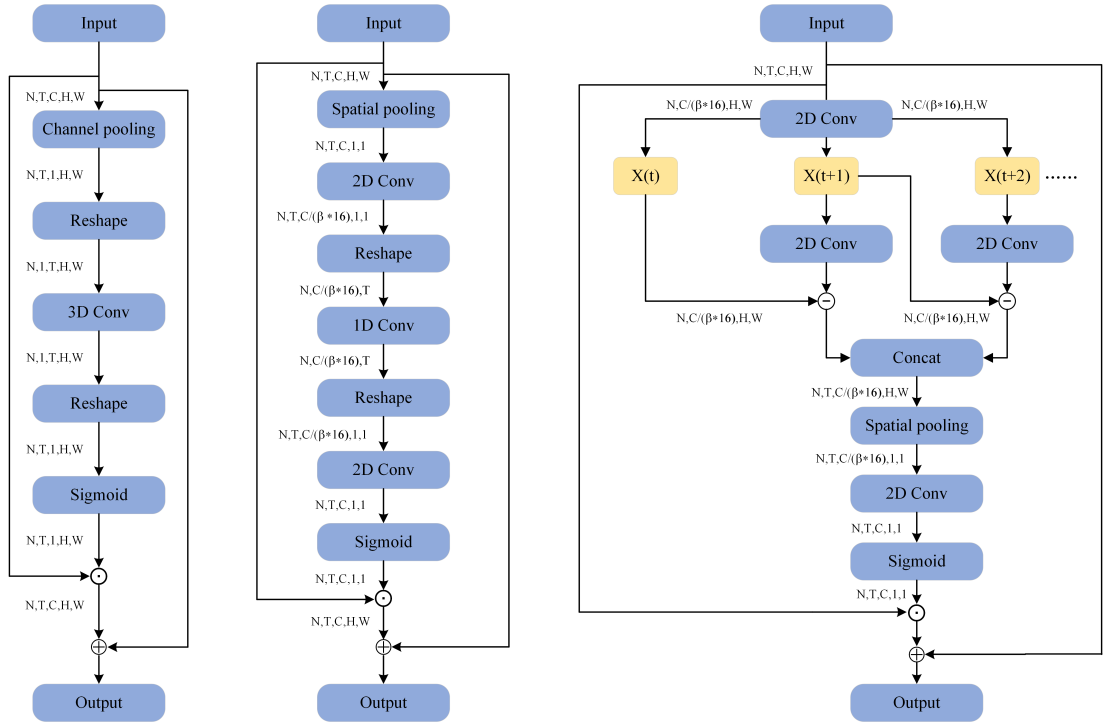

Fig. 3. Detailed structural diagram of the 3D Action module with adjusted down-sampled channels. Here, β represents the scaling factor. When $0 < \alpha \leq 0.5$, $\beta = \alpha$, and when $0.5 < \alpha < 1$, $\beta = 1 - \alpha$. The diagram illustrates the computation of attention weights for three distinct attention channels: the Spatiotemporal Excitation (STE) module, Channel Excitation (CE) module, and Motion Excitation (ME) module, shown from left to right.

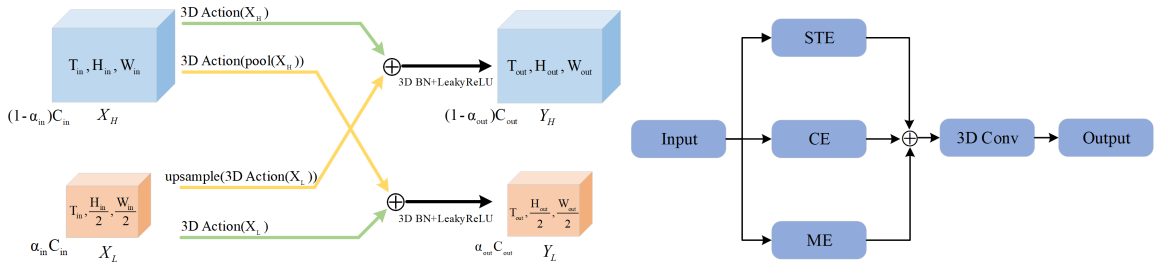

Fig. 4. The overall structural diagram of the 3D OctConv module and 3D Action module is shown in this paper. Building upon the original OctConv, we replaced the 2D convolutional module with the 3D Action module, and substituted the 2D pooling and upsampling modules with their 3D counterparts.

Subsequently, using a 1×1 2D convolution, a 1D convolution with a kernel size of 3, and reshaping operations, the tensor X_C is transformed into $X_{C\beta} \in \mathbb{R}^{N\times T\times \frac{C}{\beta\times 16}\times 1\times 1}$. Then, employing a 1×1 2D convolution, $X_{C\beta}$ is further transformed into $X^*_{C\beta} \in \mathbb{R}^{N\times T\times C\times 1\times 1}$. Finally, applying the Sigmoid function for a channel attention map $M_C \in \mathbb{R}^{N\times T\times C\times 1\times 1}$. The channel attention map is then multiplied with the initial input data and added back to the initial data to emphasize important channel features. The output of the CE module, $Y_C \in \mathbb{R}^{N\times T\times C\times H\times W}$, can be represented as:

$$Y_C = X + X \odot M_C \tag{6}$$

The Motion Excitation (ME) module utilizes a 1×1 2D convolution with a kernel size to reduce the channel number of the input data $X \in \mathbb{R}^{N\times T\times C\times H\times W}$ to $1/(\beta * 16)$ of its original, obtaining $X_M \in \mathbb{R}^{N\times T\times \frac{C}{\beta\times 16}\times H\times W}$. In the T dimension, X_M is split into the first T-1 frames $X_M^{T-1} \in \mathbb{R}^{N\times (T-1)\times \frac{C}{\beta\times 16}\times H\times W}$ and the last frame. X_M is input into a 2D convolution with a 3×3 kernel size, resulting in $X^*_M \in \mathbb{R}^{N\times T\times \frac{C}{\beta\times 16}\times H\times W}$. In the T dimension, the first frame of X^*_M is separated from the subsequent T-1 frames $X_M^{*T-1} \in \mathbb{R}^{N\times (T-1)\times \frac{C}{\beta\times 16}\times H\times W}$. Subsequently, subtracting X_M^{*T-1} from X_M^{T-1} yields $X^{**}_M \in \mathbb{R}^{N\times (T-1)\times \frac{C}{\beta\times 16}\times H\times W}$. This process can be expressed using the following formula:

$$X^{**}_M = X_M^{*T-1} - X_M^{T-1} \tag{7}$$

Continuing, by padding the final frame with zeros, we obtain $X^{**}_{ME} = [X^{**}_M(1), \cdots, X^{**}_M(T-1), 0]$. X^{**}_{ME} belongs to $\mathbb{R}^{N\times T\times \frac{C}{\beta\times 16}\times H\times W}$. Subsequently, utilizing pooling, 2D convolution with a kernel size of 1×1, and the Sigmoid function, a motion attention map M_M is derived, where M_M is in $\mathbb{R}^{N\times T\times C\times 1\times 1}$. This motion attention map is then multiplied with the initial input data and added to the original data, thereby accentuating significant motion features in the data. The output $Y_M \in \mathbb{R}^{N\times T\times C\times H\times W}$ of the ME module is

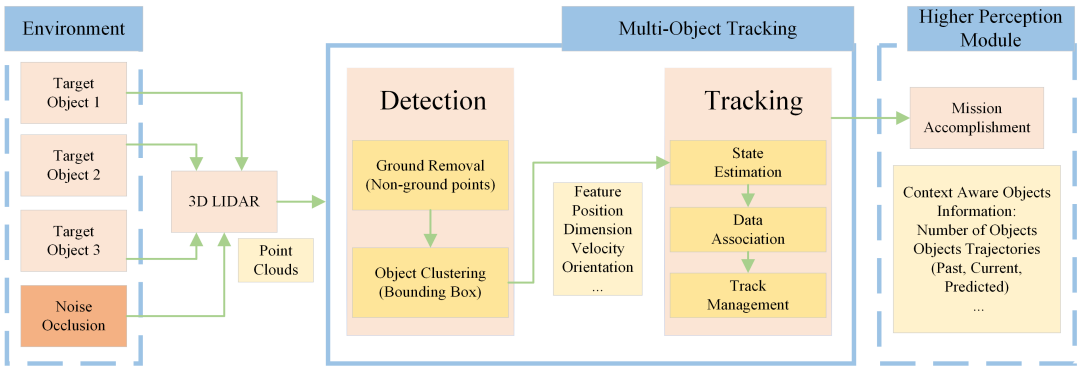

Fig. 5. Architecture of the Clustering and Tracking Module. The clustering and tracking module initially employs a slope-based channel classification method to remove ground points from the predicted point cloud. Subsequently, it employs a connected-component-based method to perform clustering on the ground-removed point cloud. Then, it utilizes TSMNet-IMM for data association and state estimation of the clustered object centers. Finally, it manages the tracking of the objects and associates the tracking trajectories with detection bounding boxes, providing output information such as the pose and trajectories of the tracked targets.

represented as:

$$Y_M = X + X \odot M_M \tag{8}$$

Finally, the outputs of the three sub-modules are summed, reshaped, and subjected to 3D convolution. The resulting data, $Y \in \mathbb{R}^{N\times C\times T\times H\times W}$, undergoes attention-weighted calculation. This can be represented mathematically as follows:

$$Y = Conv3d\,(reshape(Y_S + Y_C + Y_M)) \tag{9}$$

The 3D Octave Convolution module consists of high-frequency channels and low-frequency channels. The low-frequency channels constitute a proportion of α, while the high-frequency channels constitute $1-\alpha$. The size of images in the low-frequency channels is half that of the images in the high-frequency channels. Information exchange between high and low-frequency channels is achieved through upsampling and downsampling. The output Y_H of the high-frequency channels is obtained by adding the data output from the 3D Action Attention module applied to the input X_H of the high-frequency channels and the data output from the 3D Action Attention module applied to the input of the low-frequency channels, followed by upsampling. Batch normalization and LeakyReLU activation function are then applied to the result. Similarly, the output Y_L of the low-frequency channels is obtained by adding the data output from the 3D Action Attention module applied to the input X_L of the low-frequency channels and the data output from the downsampling and 3D Action Attention module applied to the input of the high-frequency channels, followed by batch normalization and LeakyReLU activation function.

This paper begins by standardizing the input feature tensor of size $(N, 1, P, H, W)$, which includes P-frame range images with a channel number of 1, batch size of *N*, height of *H*, and width of *W*. This tensor is then fed into an encoder composed of TCNet, 3D Octave convolution, and 3D Action attention modules. Inspired by ActionNet [18] and OctConv [19], the standardized feature tensor is first passed through a 3D convolutional input layer with $(1-\alpha)*C$ kernels, resulting in a feature tensor of size $(N, (1-\alpha)*C, P, H, W)$. Next, the feature tensor is inputted into the next encoder level. For the *i-th* encoder level, the input consists of high-frequency feature tensors processed by the 3D Octave convolution module, with a size of $(N, (1-\alpha)*C_i, P_i, H_i, W_i)$, and low-frequency feature tensors with a size of $(N, \alpha*C_i, P_i, \frac{H_i}{2}, \frac{W_i}{2})$. The 3D Action module is then employed to extract spatio-temporal, channel, and motion features from the input high-frequency and low-frequency feature tensors. This process results in high-frequency downsampled feature tensors of size $(N, (1-\alpha)*C_{i+1}, P_i - t_i, \frac{H_i}{h_i}, \frac{W_i}{w_i})$ and low-frequency downsampled feature tensors of size $(N, \alpha*C_{i+1}, P_i - t_i, \frac{H_i}{2h_i}, \frac{W_i}{2w_i})$. Here, h_i and w_i represent spatial downsampling factors, and t_i denotes the reduced temporal dimension.

To anticipate range images for future frames (*F* frames), the decoder conducts an upsampling operation on the downsampled feature tensor, resulting in a size of $(N, 2, F, H, W)$. The decoder essentially mirrors the structure of the encoder. While the number of anticipated future range images remains constant within the encoder-decoder framework, extending autoregressive prediction for a longer duration is achievable by employing the predicted range images as input tensors. In this study, the initial input feature tensor of size $(N, 1, T, H, W)$ undergoes processing through a 3D convolutional layer, yielding a feature tensor of size $(N, (1-\alpha)*C, T, H, W)$. Subsequently, the encoder-decoder architecture, comprising 3D Action modules, 3D Octave Convolution modules, and TCNet, is applied. The feature tensor from the 3D convolutional layer is fed into the encoder-decoder, which produces a feature tensor of size $(N, (1-\alpha)*C, F, H, W)$. Lastly, the feature tensor generated by the encoder-decoder undergoes processing through a 3D convolutional layer, resulting in a feature tensor of size $(N, 2, F, H, W)$. This outputted feature tensor encompasses two channels: one channel predicts the future range images, while the other channel determines the likelihood of a point being valid within the range image. By retaining points with a probability exceeding 0.5, a range image containing exclusively valid points is obtained.

C. Point Cloud Clustering and Tracking

To utilize the predicted point cloud in downstream tasks, this paper performs clustering and tracking on the predicted point

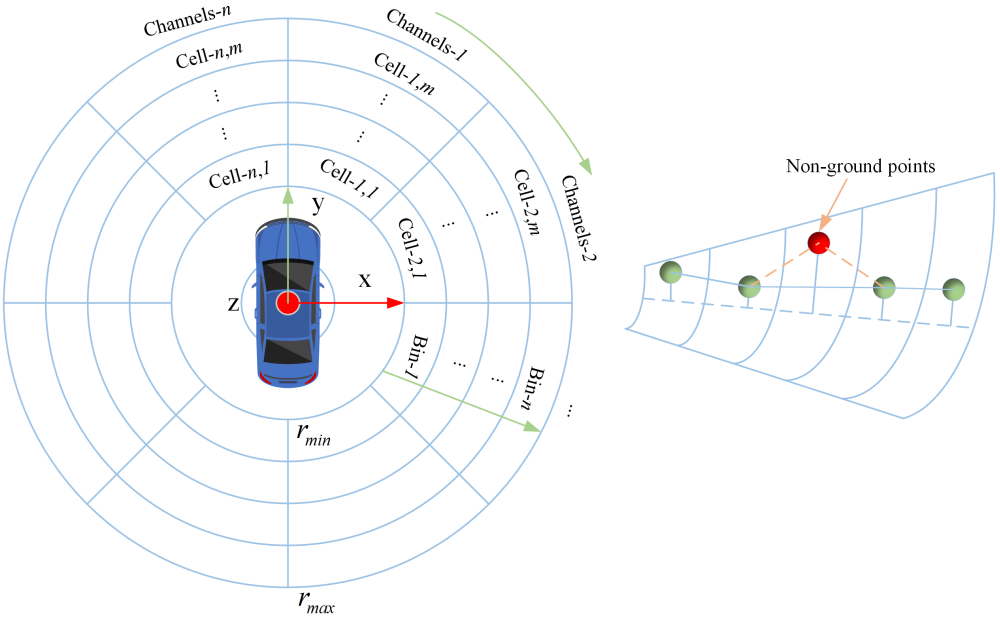

Fig. 6. Channel classification method based on slope, the left side shows the polar grid map, and the right side shows the ground and non-ground points classified.

cloud to obtain the future poses and trajectories of clustered objects. The clustering and tracking module first removes the ground from the predicted 3D point cloud, then performs clustering on the ground-removed 3D point cloud, and finally tracks the clustered objects, obtaining both the future poses and 3D trajectories of these objects. The architecture of the clustering and tracking module is illustrated in Fig. 5.

In this module, a slope-based channel classification method [36] is employed to distinguish ground points from non-ground points. The slope-based channel classification method initially utilizes the following formula:

$$\begin{aligned} \text{channels}\,(p_i) &= \frac{arctan\,(y_i, x_i)}{2\pi} \\ \text{bin}\,(p_i) &= \frac{\sqrt{{x_i}^2 + {y_i}^2} - r_{\min}}{2\pi} \end{aligned} \tag{10}$$

LiDAR point clouds $p_i = \{x_i, y_i, z_i, I_i\}$ are divided into different polar coordinate cells as shown in Fig. 6. $r_{\min}$ is the radius within which the reflection of the ego vehicle can no longer be seen, while $r_{\max}$ is determined by the effective range of the sensor. The local lowest points of the cells are compared to determine the ground height range $[T_{h\,\min}, T_{h\,\max}]$. Subsequently, the slope T_s between the local lowest points of adjacent cells is compared with the slope threshold T_{slope}, and the height difference T_h between the local lowest point and other points within the cell is compared with the height difference threshold T_{hdiff} to distinguish between ground and non-ground points. Ground points, depicted in green in Fig. 6, and non-ground points, depicted in red, are determined accordingly. Finally, better ground estimation is achieved through consistency checks and median filtering [36], resulting in a more effective differentiation between ground and non-ground points.

The distinguished non-ground points are clustered using a connected-component-based method [36]. In the connected-component-based method, non-ground point clouds are initially divided into different grid cells, with each cell having two initial states, either unoccupied (0) or occupied (-1). Then, one of the grid cells is chosen as the central cell, and a clustering ID counter is incremented. Subsequently, the occupancy status of the current central cell and all adjacent cells is checked, and occupied cells are labeled with the current clustering ID. This process is repeated for other grid cells until all non-empty clusters are assigned IDs, completing the point cloud clustering. The clustering results are shown in Fig 7.

The center points of clustered objects are tracked using the TSMNet-IMM method based on IMM-UKF-JPDAF [36], which is a ”coupled” filter. During tracking, an Interactive Multiple Model (IMM) is initially used to describe tracking objects with different motion models. IMM can adaptively switch between multiple predefined motion models to better accommodate the diverse motion characteristics of targets. Next, an Unscented Kalman Filter (UKF) is used to estimate the nonlinear motion states of tracking objects, providing more accurate predictions of target positions and velocities. To address data association problems in multi-object tracking, a Joint Probabilistic Data Association Filter (JPDAF) is employed. This method can handle dense tracking objects and estimate their association probabilities, effectively performing data association. Finally, tracking objects are managed. During tracking management, the LSTM-predicted future vehicle velocity and orientation are used to classify the maturity and dynamic or static behavior of tracking objects. Maturity and tracks with lifetimes exceeding predefined thresholds are associated with detected bounding boxes, providing information on the pose, trajectories, and other details of the tracking targets. This enriches the semantic information of tracking objects for planning and decision-making tasks.

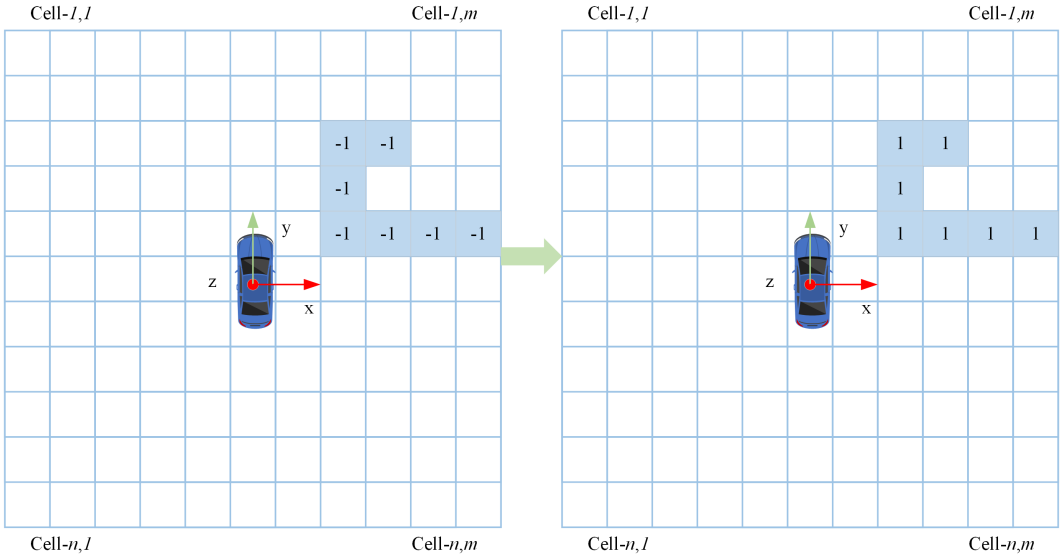

Fig. 7. Clustered result diagram. The left side represents the division of occupied and unoccupied grids, while the right side represents the result after assigning IDs to the occupied grids.

IV. EXPERIMENTS

This paper presents an annotation-free method for pose and trajectory prediction, and its efficacy is substantiated through a series of experiments conducted on a computer equipped with a CPU configuration featuring an i7-12700 processor and a GPU configuration utilizing the GeForce RTX 3080 (10GB). In Section IV-A, we introduce the loss function and the associated hyperparameters for training. In Section IV-B, we discuss the performance metrics related to experiments. In Section IV-C, we validate the ability of the self-supervised neural network, TSMNet, which is constructed upon TCNet, 3D Action attention modules, and 3D Octave convolutions as elucidated in this paper. It is demonstrated that TSMNet excels at extracting spatiotemporal information from the input past point cloud sequence, subsequently using this information to predict future point cloud sequences with superior accuracy in comparison to previous point cloud prediction methods.

Section IV-D underscores the robust performance of the point cloud prediction methodology introduced in this paper, particularly in unknown environments. Notably, it operates at a prediction speed surpassing the frame rate of typical 3D LiDAR sensors, which is 10 Hz. In Section IV-E, we analyze and compare the influence of different modules on the model's overall performance. Finally, Section IV-F involves the application of clustering and tracking techniques to the predicted point cloud sequences, yielding valuable object pose and future trajectory information.

A. Loss function and hyperparameters

The data in this paper is divided into a sequence of samples consisting of P-frame past range images and F-frame future range images, with subsequent samples separated by one frame. Sequence samples are used for model training and testing. The network in this paper has multiple losses, including range image loss, mask loss, and Chamfer Distance(CD) loss. The average range image loss $L_{R,t}$ for future frame t is calculated using the predicted range image$(\hat{r}_{i,j})_t \in \mathbb{R}^{H\times W}$ and the true range image $(r_{i,j})_t \in \mathbb{R}^{H\times W}$ through an *L1* loss function. The range image loss function is as follows:

$$L_{\mathrm{R},t} = \frac{1}{HW}\sum_{i,j}\Delta r_{i,j}, \text{ with } \Delta r_{i,j} = \begin{cases} \|\hat{r}_{i,j} - r_{i,j}\|_1 & , \text{ if } r_{i,j} > 0 \\ 0 & , \text{ otherwise} \end{cases} \tag{11}$$

The average mask loss $L_{M,t}$ for the future frame at time t is calculated based on the predicted probabilities of points being valid in the predicted range image $(\hat{m}_{i,j})_t \in \mathbb{R}^{H\times W}$ and the actual binary values indicating whether the points in the true range image are valid or not, denoted as $y_{i,j}$. This loss is computed using the *BCEWithLogitsLoss* loss function, where $y_{i,j}$ can take only two values: 0, representing invalid points, and 1, representing valid points. The mask loss function is defined as follows:

$$L_{\mathrm{M},t} = \frac{1}{HW}\sum_{i,j} -y_{i,j}\log\left(\hat{m}_{i,j}\right) - \left(1-y_{i,j}\right)\log\left(1-\hat{m}_{i,j}\right) \tag{12}$$

The range image loss and mask loss exclusively focus on the predicted 2D range image, without considering the reconstructed 3D point cloud. To enable a fair comparison with techniques relying on 3D point clouds, such as MoNet [29], this study introduces the CD as an evaluation metric for assessing the similarity between the reconstructed 3D point cloud and the ground truth point cloud. Specifically, at future frame t, the CD loss $L_{CD,t}$ is computed by projecting the range image, which includes only valid points. This is denoted as $\hat{S}_t = \left\{\hat{p} \in \mathbb{R}^3\right\}, \left|\hat{S}_t\right| = N$. Additionally, the ground truth point cloud is represented as $S_t = \left\{p \in \mathbb{R}^3\right\}, |S_t| = M$. This process is calculated as follows:

$$L_{\mathrm{CD},t} = \frac{1}{N}\sum_{\hat{\boldsymbol{p}}\in\hat{s}_t}\min_{\boldsymbol{p}\in s_t}\|\hat{\boldsymbol{p}}-\boldsymbol{p}\|_2^2 + \frac{1}{M}\sum_{\boldsymbol{p}\in s_t}\min_{\hat{\boldsymbol{p}}\in\hat{s}_t}\|\hat{\boldsymbol{p}}-\boldsymbol{p}\|_2^2 \tag{13}$$

Due to the extensive computation required for the CD, its calculation is relatively time-consuming. Utilizing only 2D image-based loss functions $L_{R,t}$ and $L_{M,t}$ is approximately

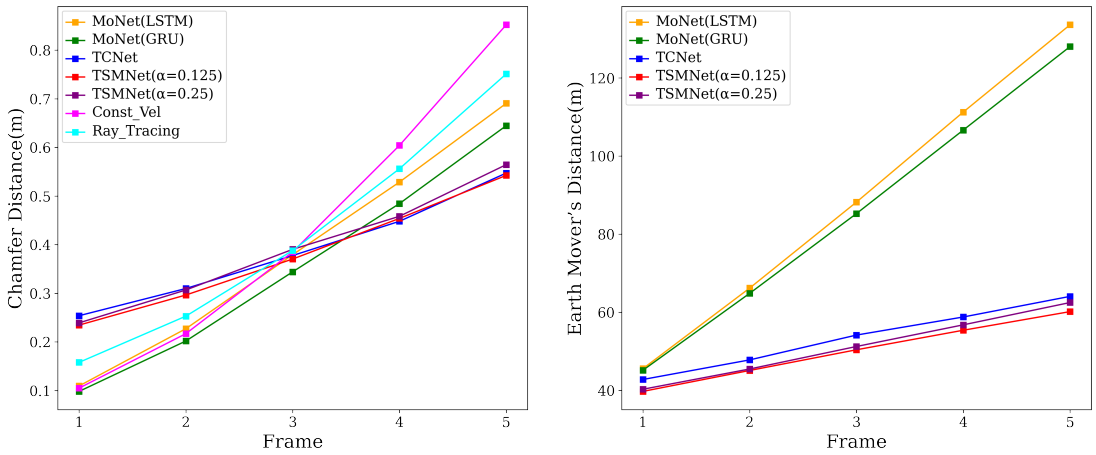

Fig. 8. The comparison of CD and EMD among different methods.

2.5 times faster than employing the combination of $L_{R,t}$, $L_{M,t}$, and $L_{CD,t}$ loss functions. Therefore, in this study, we initially pre-train the network using $L_{R,t}$ and $L_{M,t}$, and subsequently fine-tune the trained model by incorporating $L_{CD,t}$. For the current time step, denoted as t=T, the average loss over the future F frames is defined as follows:

$$L = \frac{1}{F} \sum_{t=T+1}^{T+F} L_{\mathrm{R,t}} + \alpha_{\mathrm{M}} L_{\mathrm{M,t}} + \alpha_{CD} L_{CD,t} \tag{14}$$

The adjustable weighting parameters, α_M and α_{CD}, play a crucial role in the training process. During the pre-training phase, we set $\alpha_M = 1.0$ and $\alpha_{CD} = 0.0$. However, in the fine-tuning stage, we adjust the values to $\alpha_M = 1.0$ and $\alpha_{CD} = 1.0$, as this configuration has demonstrated the best results. To optimize the network parameters, we utilize the Adam optimizer with a learning rate of 10^{-3}. Gradients are accumulated over 16 samples before updating the parameters. Additionally, the learning rate undergoes exponential decay, with the learning rate for the next round being 0.99 times the learning rate for the previous round. Typically, the entire model converges in fewer than 50 training rounds.

B. Experimental Performance Metrics

In the experiment section, relevant performance metrics include CD, Earth Mover's Distance (EMD), MOTA, and MOTP. Among these, CD and EMD are crucial indicators for assessing point cloud prediction. The calculation formula for CD is presented as $L_{CD,t}$ in Section III-D, while the formula for EMD is as follows:

$$EMD = \min_{\phi:\hat{p}_t \to p_t} \frac{1}{N} \sum_{\hat{x}^j \in \hat{p}_j} \|\hat{x} - \phi(\hat{x})\| \tag{15}$$

Where $\hat{p}_t \in \mathbb{R}^{N\times 3}$ represents the predicted point cloud, and $p_t \in \mathbb{R}^{N\times 3}$ represents the ground truth point cloud. The function $\phi : \hat{p}_t \to p_t$ is a bijective mapping from $\hat{p}_t$ to p_t. MOTA and MOTP are comprehensive performance metrics for evaluating the tracker, and their calculation formulas are as follows:

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDSW_t)}{\sum_t GT_t} \tag{16}$$

Where FN_t refers to the number of real trajectory points in frame t that were not matched, FP_t refers to the number of incorrectly predicted trajectory points in frame t, $IDSW_t$ refers to the number of times the tracking ID changes in frame t, and GT_t refers to the true number of objects present in frame t.

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{17}$$

Where $d_{t,i}$ represents the overlap ratio between the bounding box detected for target i at frame t and the ground truth bounding box, and c_t is the count of successfully matched predicted trajectory points with the true trajectory points at frame t.

C. Prediction Results

This study adopts a similar experimental setup to MoNet [29]. The experimentation leverages the KITTI Odometry dataset, focusing on predicting the subsequent 5 frames of point clouds from the preceding 5 frames captured by the LiDAR, operating at a rate of 10 Hz. The KITTI Odometry dataset encompasses a total of 22 sequences. For this research, sequences 0-5 are allocated for the training set, sequences 6-7 for the validation set, and sequences 8-10 for the test set.

TABLE I
COMPARISON OF CD(M) ON THE TEST SET FOR DIFFERENT METHODS.

Prediction Step	MoNet (LSTM)	MoNet (GRU)	TCNet	TSMNet ($\alpha = 0.125$)	TSMNet ($\alpha = 0.25$)	Const. Vel. Baseline	Ray Tracing Baseline
1	0.110	0.098	0.254	0.235	0.239	0.105	0.158
2	0.227	0.202	0.310	0.296	0.307	0.217	0.253
3	0.379	0.344	0.378	0.370	0.390	0.385	0.388
4	0.529	0.484	0.448	0.454	0.458	0.604	0.556
5	0.691	0.645	0.547	0.542	0.564	0.852	0.751
Mean	0.387	0.355	0.387	0.380	0.392	0.433	0.421

We conducted a comparative analysis of TSMNet, TCNet [16], MoNet, as well as the constant velocity baseline and ray tracing baseline obtained using SLAM [37] mentioned in

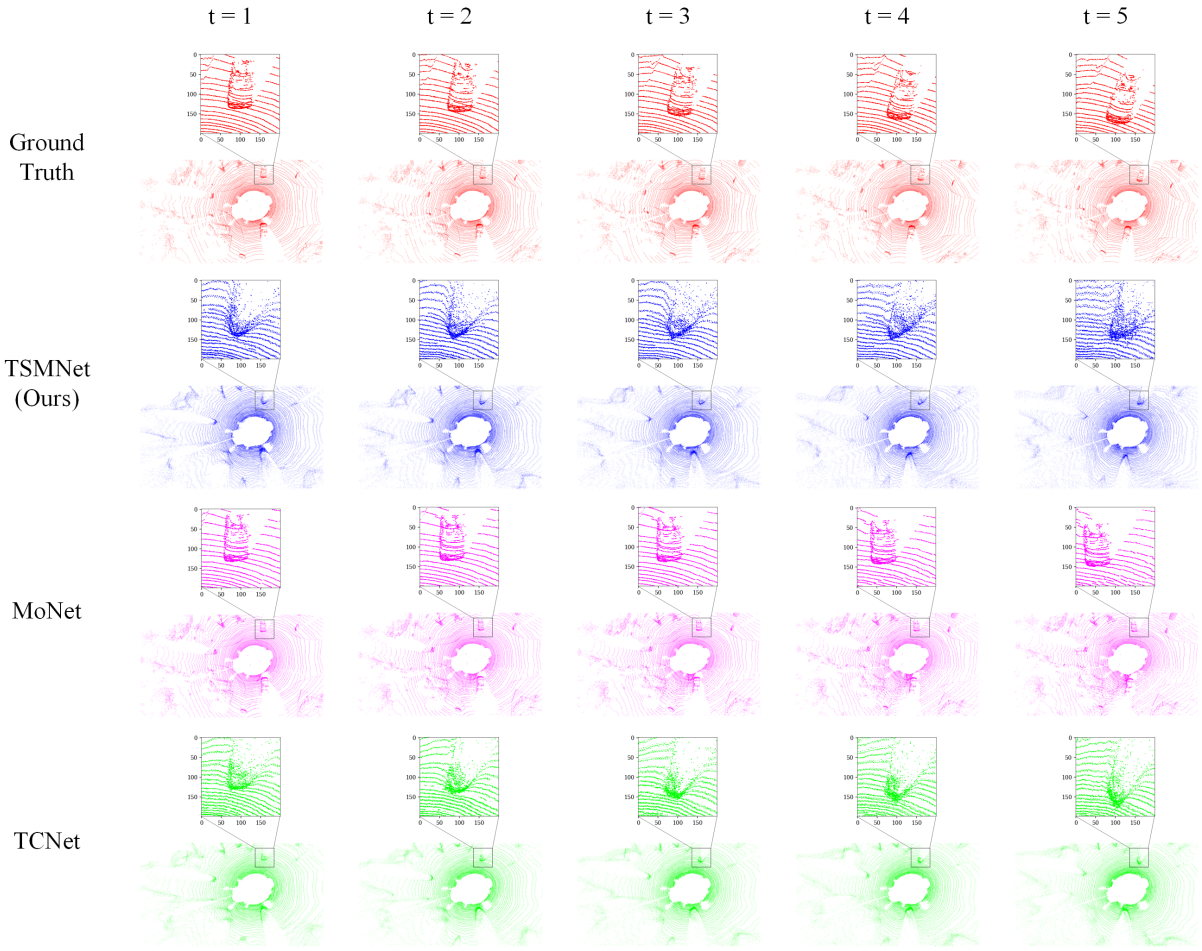

Fig. 9. Visualization of point cloud predictions by TSMNet, TCNet, and MoNet on the KITTI Odometry dataset. To visualize the best results, TSMNet made predictions with a low-frequency channel $\alpha = 0.125$, and MoNet used the GRU method for predictions. Zooming in on the highlighted region in the figure reveals that the point cloud predicted by TSMNet is relatively close in position and orientation to the real point cloud but has significant shape differences. On the other hand, the point cloud predicted by MoNet closely resembles the shape of the real point cloud but has significant differences in orientation and position. TCNet's predicted point cloud shape is similar to TSMNet, but the position and orientation of the predicted point cloud are slightly less accurate than TSMNet.

[16], based on the CD metric using the test dataset, as well as a comparative analysis of the EMD of TSMNet, TCNet, and MoNet. EMD exhibits greater sensitivity to the local intricacies and density distribution of point clouds compared to CD [38]. It's worth noting, however, that the computation of EMD demands higher memory resources and computational power. Therefore, it was computed on a randomly sampled 16,384 points, and the comparison results are shown in Table I and Table II. From the data in Table I and Table II, it can be observed that when TSMNet's low-frequency channel proportion $\alpha = 0.125$, $\alpha = 0.25$, TSMNet exhibits lower average EMD losses compared to TCNet, MoNet(LSTM), and MoNet(GRU). This indicates that the proposed method in this paper predicts point cloud distributions closer to real point clouds than the other methods. When $\alpha = 0.125$, TSMNet exhibits a decrease in average CD compared to TCNet, MoNet (LSTM), the constant velocity baseline, and the ray tracing baseline but is slightly higher than MoNet(GRU). Referring to Fig. 8, it can be seen that except for TSMNet and TCNet, the CD and EMD increase rapidly as the prediction length increases. Additionally, TSMNet exhibits smaller CD and EMD than TCNet when considering the low-frequency channel proportion $\alpha = 0.125$. Therefore, our proposed method outperforms the other methods in long-term point cloud prediction.

TABLE II
COMPARISON OF EMD(M) ON THE TEST SET FOR DIFFERENT METHODS.

Prediction Step	MoNet (LSTM)	MoNet (GRU)	TCNet	TSMNet ($\alpha = 0.125$)	TSMNet ($\alpha = 0.25$)
1	45.575	45.162	42.801	39.749	40.298
2	66.206	64.866	47.818	45.108	45.490
3	88.223	85.283	54.175	50.409	51.251
4	111.275	106.666	58.802	55.409	56.791
5	133.623	128.041	64.080	60.179	62.501
Mean	88.981	86.003	53.535	**50.171**	**51.266**

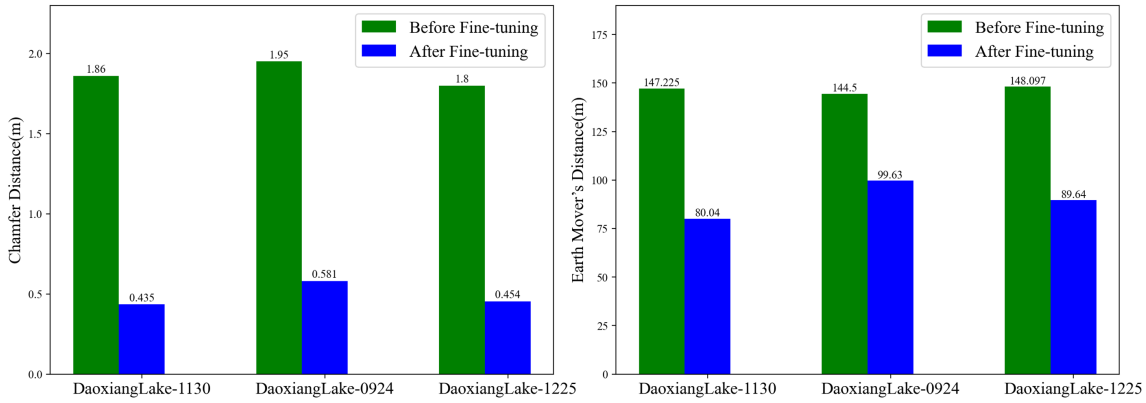

Fig. 10. The results of TSMNet model tested on the Apollo-DaoxiangLake dataset. The left side shows the comparison of CD before and after fine-tuning, while the right side shows the comparison of EMD before and after fine-tuning.

To visually represent the CD losses and EMD losses of different methods at different time steps, this paper compares the CD and EMD of these methods on the test set from the first frame to the fifth frame in Fig. 8.

To offer a more intuitive comparison of the predictive capabilities of various point cloud prediction methods, this paper presents exemplar point cloud prediction outcomes for different methodologies using the KITTI Odometry dataset, as illustrated in Fig. 9. The chosen instances encompass point cloud predictions occurring during vehicular maneuvers involving turns. It is noteworthy that during such turning maneuvers, the coordinate system of the LiDAR sensor experiences rotation, thereby rendering point cloud prediction in this context a more accurate representation of the performance of point cloud prediction algorithms. In the figure, from left to right, five successive future point clouds at t=1 to t=5 are exhibited, with a particular focus on the section encapsulating the point cloud of the vehicle.

D. Model Generalization Ability

To validate the generalization ability of the TSMNet model on other datasets, this study utilize the fine-tuned model on the KITTI Odometry dataset to conduct tests on frames 46146 to 58577, frames 60306 to 73241, and frames 40924 to 49504 of the Apollo-DaoxiangLake dataset [39]. The results of the tests are shown by the green bars in Fig 10.

The TSMNet model was tested on three datasets from Apollo-DaoxiangLake, yielding average CD of 1.86 m, 1.95 m, and 1.80 m, and average EMD of 147.225, 144.500, and 148.097, respectively, which significantly deviated from the results shown in Table 1. This discrepancy arises from the fact that the Apollo-DaoxiangLake dataset was collected in China and differs substantially from the KITTI Odometry dataset collected in Germany. Therefore, this paper conducted single-round fine-tuning of the TSMNet model on three datasets from Apollo-DaoxiangLake, specifically on the frames ranging from 8063 to 38142, 8177 to 47600, and 6710 to 32356 for datasets DaoxiangLake-1130, DaoxiangLake-0924, and DaoxiangLake-1225, respectively. The final results showed average CD of 0.435 m, 0.581 m, and 0.454 m, and average EMD of 80.04, 99.63, and 89.64. The outcomes of single-round fine-tuning demonstrate the generalization ability and advantages of TSMNet through self-supervised training.

Regarding the model's inference time, when the TSMNet model with a low-frequency channel ratio of $\alpha = 0.125$ is used, it takes an average of 38 milliseconds (26 Hz) to predict the future 5 frames of point clouds on a computer with a CPU configured as i7-12700 and a GPU configured as GeForce RTX 3080 (10GB). When $\alpha = 0.25$, it takes an average of 33 milliseconds (30 Hz) to predict the future 5 frames of point clouds, which is faster than the typical frame rate of 3D LiDAR sensors (10 Hz).

E. Ablation Experiments

In this experiment, the paper analyzes the impact of incorporating different modules on the model's performance. This experiment evaluates the model's performance on the validation set during the training period, and the evaluation results are shown in Fig. 11.

The paper introduces the 3D Action Attention module into TCNet, which enhances the model's ability to extract spatiotemporal and motion information from the input point cloud, thereby improving the model's performance. However, the inclusion of the attention module results in a relatively high computational workload for the model. Additionally, the paper introduces High-Low Frequency Convolution 3D OctConv into TCNet. This convolution separates the high-frequency and low-frequency channels, with the image size in the low-frequency channel being half that of the high-frequency channel. There is information exchange between the high and low-frequency channels. Using High-Low Frequency Convolution 3D OctConv as a replacement for the regular 3D Convolution (Conv3D) can reduce the model's computational workload. However, when the low-frequency channels of 3D OctConv account for a higher proportion, denoted as α, it may decrease the performance of the model. By combining TCNet, 3D OctConv, and 3D Action, the model's performance can be enhanced while reducing computational complexity. For example, when the low-frequency channel ratio α is set to 0.125 or 0.25, it means that 12.5% or 25% of the

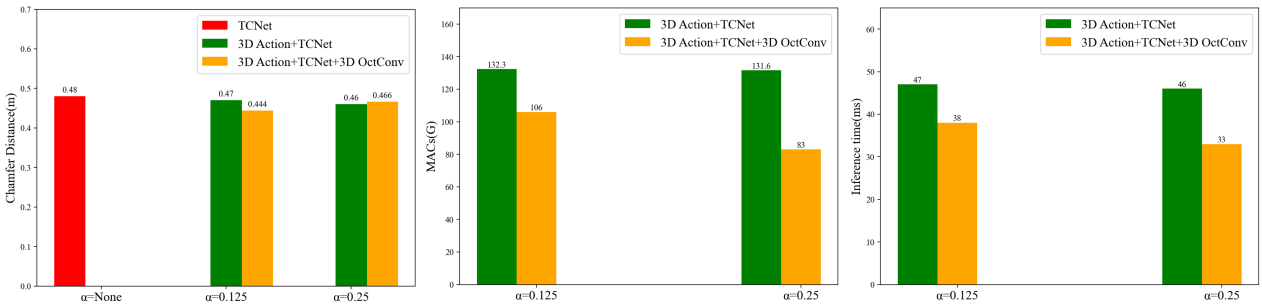

Fig. 11. The performance of different modules. Left shows the impact of different modules on CD loss, the middle illustrates the impact of different modules on computational complexity, and the right indicates the effect of different modules on model inference time.

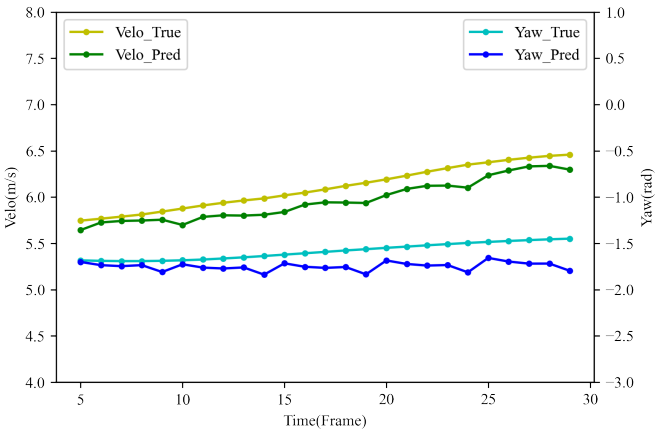

Fig. 12. A comparison between the predicted ego vehicle speed and yaw angle for the next 5 frames using LSTM and the ground truth ego vehicle speed and yaw angle. From the figure, it can be observed that the predicted values closely match the ground truth values, indicating good prediction performance.

current channel's feature map size is reduced by half, while the remaining 87.5% or 75% of the channel's feature map size remains unchanged. Because some channels' feature map sizes are reduced by half, the computational workload of the convolution operation decreases, thus reducing the model's computational complexity and accelerates the inference time. Moreover, due to the information exchange between the high and low-frequency channels in 3D OctConv, and the improved effect brought by the 3D Action module when the low-frequency channel ratio is higher, the performance of the TSMNet model is better than TCNet when the low-frequency channel ratio is not too high (12.5%, 25%).

F. Clustering and Tracking Results

To further validate the effectiveness of point cloud prediction by applying the predicted point cloud to downstream tasks, this paper conducts clustering and tracking on the predicted point cloud after point cloud prediction. The multi-object tracking algorithm used in this paper is TSMNet-IMM, and the tracker's performance is evaluated using the MOT16 benchmark method proposed by Milan et al. The tracker's performance on the KITTI RAW dataset is shown in Table III.

TABLE III
PERFORMANCE METRICS OF TSMNET-IMM TRACKER IN DIFFERENT SCENARIOS

Dataset	FP	FN	Frag	IDSW	MOTA	MOTP
0001	2	8	0	2	91.61%	94.25%
0002	1	2	0	0	93.33%	85.10%
0005	7	34	8	8	89.73%	91.87%
0009	33	172	99	41	82.59%	89.64%
0013	2	2	0	0	96.04%	97.05%
0017	0	14	8	2	80.95%	93.86%
0018	2	37	27	2	79.08%	95.22%
0048	0	12	3	1	84.34%	93.83%
0051	5	59	13	7	81.36%	88.38%
0057	13	66	53	5	82.17%	80.90%
Avg	6.5	40.6	21.1	6.8	86.12%	91.01%

We selected data from 10 different scenes in KITTI RAW, including 0001, 0002, 0005, 0009, 0013, 0017, 0018, 0048, 0051, and 0057, for multi-object tracking evaluation. These

TABLE IV
COMPARISON OF ONLINE TRACKING ALGORITHMS

Method	MOTA	MOTP
TSMNet-IMM	86.12%	**91.01%**
TuSimple	86.62%	83.97%
RRC-IIITH	84.24%	85.73%
IMMDP	83.04%	82.74%
DuEye	80.64%	83.52%
JCSTD	80.57%	81.81%
wan	78.07%	82.83%
CCF-MOT	77.08%	78.36%
MDP	76.59%	82.10%
SCEA	75.58%	79.39%
CIWT	75.39%	79.25%

10 scenes comprise 2111 frames and 3387 objects. The table below lists the multi-object tracking performance of the tracker

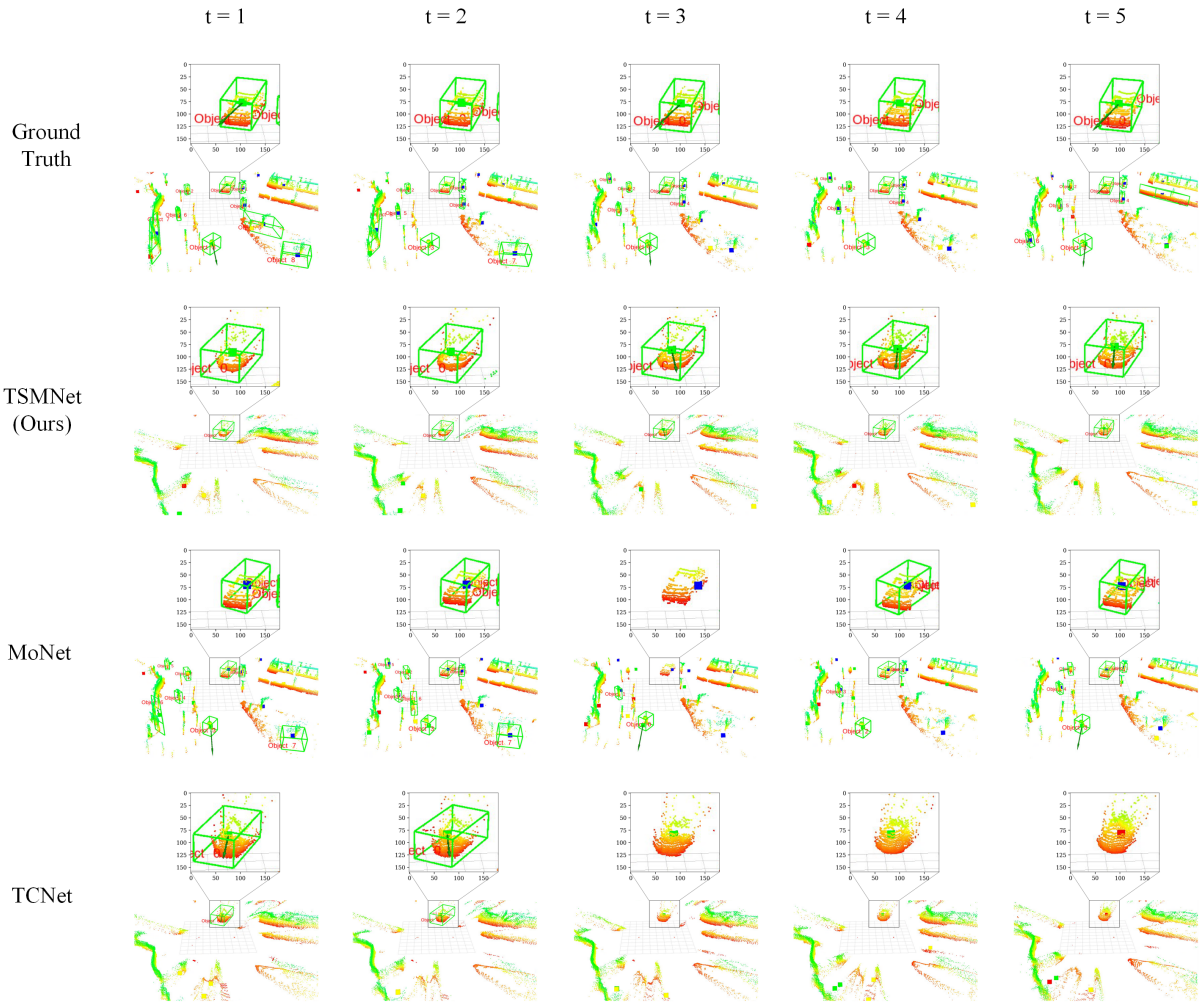

Fig. 13. Clustering and Tracking Results of Predicted Point Clouds by TSMNet, TCNet, MoNet, and Real Point Clouds. The figure provides an overview of the results for the next 5 frames (t=1 to t=5, left to right). Different-colored object centroids are shown, with each color representing a different state of the objects: green for objects in motion relative to the vehicle (dynamic track), blue for stationary or relatively stationary objects (static track), yellow for initialization tracks (indicating recently detected objects that have just entered the detection area and their tracks are not yet fully established), and red for drift tracks (indicating tracks that are about to be lost due to entering a blind zone). Tracks that meet a predefined threshold for maturity and lifespan are associated with the detected bounding boxes. The size, orientation, and tracking ID of the associated bounding boxes are displayed. Tracks that do not meet the predefined threshold are not displayed.

in each scene. MOTA and MOTP are metrics that reflect the tracker's overall performance. MOTA combines FP, FN, and IDSW to indicate the tracker's accuracy, while MOTP reflects the tracker's localization precision. From the table, it can be observed that the tracker maintains a MOTA of over 79% in different scenes, with an average MOTA of 86.12%. This suggests that the tracker exhibits high tracking accuracy across various scenes. The MOTP in different scenes also remains above 80%, with an average MOTP of 91.01%, indicating high localization precision in different scenarios. In Table IV, the multi-object tracking algorithm used in this paper, TSMNet-IMM, is also compared with the top 10 online benchmark algorithms for car tracking in the 2012 KITTI dataset, including TuSimple, RRC-IIITH, IMMDP, DuEye, JCSTD, wan, CCF-MOT, MDP, SCEA, and CIWT. From Tables III and IV, it can be seen that the tracker's overall performance has reached a relatively high level.

In the process of conducting multi-object detection and tracking, it becomes imperative to anticipate the forthcoming speed and yaw angle of the host vehicle. This enables the determination of whether the neighboring objects are at a standstill, in a state of relative rest, or undergoing relative motion in relation to the host vehicle. To accomplish this, the paper integrates an LSTM model for the prediction of the host vehicle's future speed and yaw angle. The LSTM is trained and evaluated using the KITTI RAW dataset, and the outcomes of this evaluation are depicted in Fig. 12.

After integrating the predicted point cloud results and LSTM-predicted outcomes into the point cloud clustering and tracking algorithm, Fig. 13 presents the conclusive clustering and tracking results for the subsequent 5 frames. The figure offers an enlarged perspective of the tracked objects, with particular emphasis on the object labeled with the "Object0" identifier. The object detection algorithm used in this paper is a clustering algorithm, where "Object0" simply represents the tracking ID and does not classify the objects. A com-

parison between the ground truth bounding boxes and the predicted bounding boxes in each frame is depicted in the figure, reflecting the algorithm's trajectory prediction and pose prediction capabilities. Upon close examination of the figure, it becomes evident that MoNet's predicted point cloud shape closely approximates the ground truth point cloud, thereby leading to the appearance of bounding boxes with a number and size resembling those derived from the real point cloud. Nevertheless, it is noteworthy that the orientation and position of MoNet's bounding boxes significantly deviate from the ground truth point cloud, culminating in less stable tracking outcomes.

In contrast, TSMNet generates point cloud shapes that, overall, differ substantially from the genuine point cloud. Consequently, the figure displays fewer bounding boxes compared to the ground truth point cloud. Nevertheless, for objects where the point cloud shapes closely align with reality, TSMNet's bounding box characteristics—including size, orientation, and position—exhibit remarkable similarity, thus enabling stable tracking. Comparatively, TCNet's performance in point cloud prediction falls slightly behind the approach introduced in this paper. Consequently, TCNet yields less stable tracking results when contrasted with TSMNet.

V. Conclusion

This paper introduces a novel approach for self-supervised point cloud prediction by the proposed TSMNet neural network architecture, which comprises the TCNet, 3D Action Attention Module, and 3D Octave convolution. Moreover, it incorporates a clustering and tracking module for processing the predicted point cloud, ultimately yielding object poses and future trajectories. Compared to existing methods, TSMNet excels at extracting motion information from input point clouds, primarily attributed to the 3D Action Attention Module, resulting in detailed and comprehensive predictions of forthcoming point clouds. These predicted point clouds are subsequently fed into the clustering and tracking module, facilitating the estimation of object poses and future trajectories. Notably, this approach enables the prediction of trajectories and poses in three-dimensional space without the necessity for annotated data. TSMNet's performance is rigorously evaluated across various datasets and is benchmarked against existing methodologies, consistently showcasing its superiority and versatility in uncharted environments. Nevertheless, TSMNet exhibits certain limitations, including discrepancies between predicted and ground truth point cloud shapes. Future research endeavors could be directed towards mitigating these disparities to enhance prediction precision. Additionally, there is a need for further exploration into generalizing this method across diverse environments and integrating domain-specific knowledge to achieve more accurate pose and trajectory predictions.

References

[1] Y. Liu, T. Yuan, Y. Wang, Y. Wang, and H. Zhao, "Vectormapnet: End-to-end vectorized hd map learning," in *International Conference on Machine Learning*. PMLR, 2023, pp. 22 352–22 369.

[2] Q. He, Z. Wang, H. Zeng, Y. Zeng, Y. Liu, S. Liu, and B. Zeng, "Stereo rgb and deeper lidar-based network for 3d object detection in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 24, no. 1, pp. 152–162, 2022.

[3] H. Wu, W. Han, C. Wen, X. Li, and C. Wang, "3d multi-object tracking in point clouds based on prediction confidence-guided data association," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 6, pp. 5668–5677, 2021.

[4] H. Li, H. Guan, L. Ma, X. Lei, Y. Yu, H. Wang, M. R. Delavar, and J. Li, "Mvpnet: A multi-scale voxel-point adaptive fusion network for point cloud semantic segmentation in urban scenes," *International Journal of Applied Earth Observation and Geoinformation*, vol. 122, p. 103391, 2023.

[5] Y. Li, J. Cai, Q. Zhou, and H. Lu, "Joint semantic-instance segmentation method for intelligent transportation system," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[6] K. Gao, P. Luo, J. Xie, B. Chen, Y. Wu, and R. Du, "Energy management of plug-in hybrid electric vehicles based on speed prediction fused driving intention and lidar," *Energy*, vol. 284, p. 128535, 2023.

[7] K. Gao, X. Li, B. Chen, L. Hu, J. Liu, R. Du, and Y. Li, "Dual transformer based prediction for lane change intentions and trajectories in mixed traffic environment," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[8] J. Yang, D. Chu, J. Yin, D. Pi, J. Wang, and L. Lu, "Distributed model predictive control for heterogeneous platoon with leading human-driven vehicle acceleration prediction," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[9] L. Hu, T. Lu, G. Li, X. Zhang, and H. Cai, "Automatic generation of intelligent vehicle testing scenarios at intersections based on natural driving datasets," *IEEE Transactions on Intelligent Vehicles*, 2023.

[10] Z. Meng, X. Xia, R. Xu, W. Liu, and J. Ma, "Hydro-3d: Hybrid object detection and tracking for cooperative perception using 3d lidar," *IEEE Transactions on Intelligent Vehicles*, 2023.

[11] H. Gao, D. Fang, J. Xiao, W. Hussain, and J. Y. Kim, "Camrl: A joint method of channel attention and multidimensional regression loss for 3d object detection in automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, 2022.

[12] S. Rahmani, A. Baghbani, N. Bouguila, and Z. Patterson, "Graph neural networks for intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, 2023.

[13] B. Fei, W. Yang, L. Liu, T. Luo, R. Zhang, Y. Li, and Y. He, "Self-supervised learning for pre-training 3d point clouds: A survey," *arXiv preprint arXiv:2305.04691*, 2023.

[14] J. Fang, J. Qiao, J. Bai, H. Yu, and J. Xue, "Traffic accident detection via self-supervised consistency learning in driving scenarios," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 7, pp. 9601–9614, 2022.

[15] J. James, "Sybil attack identification for crowdsourced navigation: A self-supervised deep learning approach," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 7, pp. 4622–4634, 2020.

[16] B. Mersch, X. Chen, J. Behley, and C. Stachniss, "Self-supervised point cloud prediction using 3d spatio-temporal convolutional networks," in *Conference on Robot Learning*. PMLR, 2022, pp. 1444–1454.

[17] X. Weng, J. Wang, S. Levine, K. Kitani, and N. Rhinehart, "Inverting the pose forecasting pipeline with spf2: Sequential pointcloud forecasting for sequential pose forecasting," in *Conference on robot learning*. PMLR, 2021, pp. 11–20.

[18] Z. Wang, Q. She, and A. Smolic, "Action-net: Multipath excitation for action recognition," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 13 214–13 223.

[19] Y. Chen, H. Fan, B. Xu, Z. Yan, Y. Kalantidis, M. Rohrbach, S. Yan, and J. Feng, "Drop an octave: Reducing spatial redundancy in convolutional neural networks with octave convolution," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 3435–3444.

[20] R. Chandra, T. Guan, S. Panuganti, T. Mittal, U. Bhattacharya, A. Bera, and D. Manocha, "Forecasting trajectory and behavior of road-agents using spectral clustering in graph-lstms," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4882–4890, 2020.

[21] S. Carrasco, D. F. Llorca, and M. Sotelo, "Scout: Socially-consistent and understandable graph attention network for trajectory prediction of vehicles and vrus," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 1501–1508.

[22] S. H. Park, G. Lee, J. Seo, M. Bhat, M. Kang, J. Francis, A. Jadhav, P. P. Liang, and L.-P. Morency, "Diverse and admissible trajectory forecasting through multimodal context understanding," in *Computer Vision–ECCV*

2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XI 16. Springer, 2020, pp. 282–298.
[23] G. Wang, X. Wu, Z. Liu, and H. Wang, "Hierarchical attention learning of scene flow in 3d point clouds," *IEEE Transactions on Image Processing*, vol. 30, pp. 5168–5181, 2021.
[24] Y. Shi and K. Ma, "Safit: Segmentation-aware scene flow with improved transformer," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 10 648–10 655.
[25] X. Gu, C. Tang, W. Yuan, Z. Dai, S. Zhu, and P. Tan, "Rcp: Recurrent closest point for scene flow estimation on 3d point clouds," *arXiv preprint arXiv:2205.11028*, 2022.
[26] G. Puy, A. Boulch, and R. Marlet, "Flot: Scene flow on point clouds guided by optimal transport," in *European conference on computer vision*. Springer, 2020, pp. 527–544.
[27] Y. Kittenplon, Y. C. Eldar, and D. Raviv, "Flowstep3d: Model unrolling for self-supervised scene flow estimation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 4114–4123.
[28] B. Li, C. Zheng, S. Giancola, and B. Ghanem, "Sctn: Sparse convolution-transformer network for scene flow estimation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 36, no. 2, 2022, pp. 1254–1262.
[29] F. Lu, G. Chen, Z. Li, L. Zhang, Y. Liu, S. Qu, and A. Knoll, "Monet: Motion-based point cloud prediction network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 13 794–13 804, 2021.
[30] T. Khurana, P. Hu, D. Held, and D. Ramanan, "Point cloud forecasting as a proxy for 4d occupancy forecasting," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 1116–1124.
[31] T. Zou, G. Chen, F. Lu, Z. Li, S. Qu, A. Knoll, and C. Jiang, "Rpp-net: Rigid constrained point cloud prediction network," *IEEE Transactions on Intelligent Transportation Systems*, 2023.
[32] T. Yin, X. Zhou, and P. Krahenbuhl, "Center-based 3d object detection and tracking," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 11 784–11 793.
[33] F. Ruppel, F. Faion, C. Gläser, and K. Dietmayer, "Transformers for multi-object tracking on point clouds," in *2022 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2022, pp. 852–859.
[34] Y. Chen, J. Liu, X. Zhang, X. Qi, and J. Jia, "Voxelnext: Fully sparse voxelnet for 3d object detection and tracking," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 21 674–21 683.
[35] H. Wei, E. Xu, J. Zhang, Y. Meng, J. Wei, Z. Dong, and Z. Li, "Bushnet: Effective semantic segmentation of bush in large-scale point clouds," *Computers and Electronics in Agriculture*, vol. 193, p. 106653, 2022.
[36] A. Arya Senna Abdul Rachman, "3d-lidar multi object tracking for autonomous driving: Multi-target detection and tracking under urban road uncertainties," 2017.
[37] J. Behley and C. Stachniss, "Efficient surfel-based slam using 3d laser range data in urban environments." in *Robotics: Science and Systems*, vol. 2018, 2018, p. 59.
[38] M. Liu, L. Sheng, S. Yang, J. Shao, and S.-M. Hu, "Morphing and sampling network for dense point cloud completion," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 07, 2020, pp. 11 596–11 603.
[39] Y. Zhou, G. Wan, S. Hou, L. Yu, G. Wang, X. Rui, and S. Song, "Da4ad: End-to-end deep attention-based visual localization for autonomous driving," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII 16*. Springer, 2020, pp. 271–289.

Ronghua Du was born in 1973 in Hunan province. He received the Ph.D. degree from the College of Computer Science and Technology, National University of Defense Technology, China. He has been the director of The Institute of Intelligent Transportation and Vehicle-Road Collaborative Technology of Changsha University of Science and Technology since March 2011. He is currently a Professor with the Changsha University of Science and Technology. His areas of expertise include cooperative vehicle-infrastructure systems, ITS, vehicle dynamics, and connected automated vehicles.

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.

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.

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

Linhong Liu is currently pursuing the M.S. degree with the Changsha University of Science and Technology. His research interests include intelligent transportation and Deep Reinforcement Learning.