An End-to-End ConvLSTM-based Method for Point Cloud Streaming Compression

Junhua Long, Mingjian Feng, Boyang Li, Yehua Ling, Chenghao Wu, Kai Huang, and Mingyue Cui

***Abstract*— The processing and transmission of point cloud frame sequences is an important part of the applications of 3D LiDAR. However, due to the disorderliness and irregularity of the huge amount of point cloud data collected by 3D LiDAR sensor, finding an effective method to compress the point cloud data to a small volume is an urgent problem. In this paper, we propose spatio-temporal features sensitive neural network SPCCNet with an encoder-decoder structure to compress point cloud streams. To reduce information loss in point cloud preprocessing, we propose a unique convolution method on point sets. The curvature and density information are introduced to SPCCNet to enhance the raw point cloud data. Besides, the designed ConvLSTMlm and a Squeeze-and-Excitation (SE) Block are embedded to help SPCCNet learn the effective features of point cloud sequences. Experimental results show that compared with other methods, our SPCCNet can compress point cloud data with a higher compression ratio at an acceptable noise level.**

I. Introduction

3D LiDAR is an active remote sensing equipment based on photoelectric detection, which obtains a wider view field (e.g., 360 in horizontal directions) and more direct 3D environment information. Up to now, 3D LiDAR has been widely used in detection, segmentation, planning, and other fields of intelligent vehicles [1]. A regular multi-line 3D LiDAR can provide up to millions of point cloud data. Processing such a large amount of sensor data requires enormous computing power, which brings a serious challenge to the OnBoard Units (OBUs) with limited computation resources.

The huge amount of point cloud data and its characteristics of disorderness and sparsity make it hard to compress. Fortunately, several point cloud compression methods have been proposed. Some works [2]–[4] map point clouds into 2d images and apply image compression methods to them. Transforming point clouds into images leads to information loss, therefore, researchers propose extracting features from 3d voxels or original point clouds and further applying auto-encoders to compress point clouds [5]–[7]. Recently, hand-crafted and deep learning based entropy coding methods [4], [8], [9] have shown better compression performance than other methods. However, these methods do not fully consider the point cloud streaming data and utilize the information. Therefore, a sophisticated method is needed to encode and decode point cloud data efficiently for 3D LiDAR.

In this paper, we propose an end-to-end streaming point cloud compression network (SPCCNet) for the 3D LiDAR sensor, which focuses on the point cloud geometry specifically. We adopt the structure of recurrent neural networks that can effectively process sequential data. SPCCNet mainly consists of the encoder and decoder modules, which are decoupled. The key to achieving point cloud stream compression is to learn spatio-temporal information about the point cloud stream. Therefore, in the encoding process, we provide curvature and density information to learn the characteristics of the point cloud stream better. To reduce information loss in point cloud preprocessing, we use a special convolution way on point sets. Our model focuses on handling the LiDAR frame sequences but not freeform 3D point clouds, consequently, we propose the modified ConvLSTM, named ConvLSTMlm, as the main structure. The ConvLSTMlm drops the forget gate and adds a mask learned from the previous frame for the long-term information. We also embed a SE Block [10] in the encoder to improve the weight of effective features in the compression process. Finally, based on the node-to-node correspondence, we use a combination of two losses that encode both the global geometric constraints and local similarities. In summary, our contributions are threefold:

- We propose an end-to-end Density-Curvature-Weighted Network architecture with a decoupled encoder and decoder, named SPCCNet, for 3D LiDAR to effectively compress the point cloud frame sequences.
- We modify ConvLSTM and apply it to the SPCCNet to learn spatio-temporal features. We remove the forget gate and perform a two-layer convolution on the long-term information to form a mask that helps convergence.
- We evaluate our method with different evaluation metrics on two open-source datasets. Experiments show that compared with other methods, our SPCCNet can achieve a better Peak signal-to-noise Ratio (PSNR) [11] with a higher compression ratio.

II. Related work

A. Tree Structures Encoding

Among the prior point cloud compression algorithms, tree structures, especially octrees, are one of the most common encoding methods. An octree stores point cloud data by

*This work was supported by the Guangxi Key R & D Program (No. GuikeAB24010324).

J. Long, M. Feng, B. Li, C. Wu, M. Cui, and K. Huang are with the School of Computer Science and Engineering, Sun Yat-sen University, Guangzhou 400100, China (email: {longjh7, fengmj8, liby3, wuchh9, cuimy}@mail2.sysu.edu.cn; huangk36@mail.sysu.edu.cn). (The corresponding author is M. Cui, phone: +8613726211441).

Y. Ling is with the Guangxi Transportation Science and Technology Group Co., Ltd, Nanning 530029, China (email: lingyehua@foxmail.com).

recursively partitioning the space into eight octants. The node of the octree is stored by an 8-bit symbol, which represents the occupancy of each node. The leaf nodes of the octree store the spatial coordinates of the point clouds they encompass. Draco [12] is an open-source library developed by Google for compressing and decompressing 3D geometric meshes and point clouds. The library implements the KD-tree encoding of point clouds, which is widely used by researchers. The G-PCC method [4] is the MPEG's point cloud compression standard based on octree. By introducing a binarization scheme, the G-PCC can compress the occupancy information with an optimal set of binary entropy coders.

B. Methods Based on Two-dimensional Image

Inspired by the two-dimensional image compression, many studies [2] convert 3D point cloud data into 2D format and then use various image/video compression methods to reduce the amount of data. Beemelmanns *et al.* [3] propose using common image compression methods and a recurrent neural network to compress point cloud data. MPEG V-PCC [4] is based on the previously developed video compression technology, which projects 3D point clouds onto 2D. By converting the point cloud into a set of different video sequences, V-PCC leverages existing video codecs for compressing the geometry and texture information of a dynamic point cloud.

C. 3D Data Structure Encoding

There are two mainly 3D-based ways for point cloud encoding. One is the voxel-based method [13], which divides raw point cloud data into grids and applies 3D convolutions on these voxels. The other is the point-based method [14], which directly processes the point cloud data by feeding the data into a specially designed neural network. Yan *et al.* [6] propose an auto-encoder-based network for lossy geometry point cloud compression. This method directly takes point clouds as input rather than voxel grids or image collections. Nguyen *et al.* [7] propose using neural networks to estimate the probability distribution of voxel occupancy. They use a context-based arithmetic coder to encode voxels according to the predicted probability distributions.

Until now, there have not been many thorough prior studies on streaming point cloud compression for 3D LiDAR sensors, which prompted this work. Our method focuses on the streams of LiDAR frame sequences, which is of importance in scenarios such as autonomous driving. We propose an end-to-end decoupled auto-encoder-based model to compress point cloud frame sequences, which can be applied to V2X scenarios and downstream perception tasks without hindering performance.

III. Method

Our SPCCNet architectures are shown in Fig. 1. We introduce a unique way to rearrange the point cloud data and apply convolution operations on three channels (density, curvature, and raw data). Our proposed network is an auto-encoder-based model, which uses ConvLSTMlm to better capture the temporal features. The encoder and decoder manipulate multiple ConvLSTMlm and ConvTranspose [15] layers to encode and decode the point cloud streams. Besides, we design a novel loss function that helps the network converge faster and fit better.

A. Convolutions on Point Sets

In order to fuse additional features like curvature better, we propose a special organization way of the point cloud data. Firstly, the data is stored in three 1D arrays, we sort them by the magnitude of z values. If there are two same z values, we further sort them by y. Then we transform each array into the 2D format. For example, if the size of sorted x array is (1024×1), we simply take the first 32 elements as the first row, elements 33 to 64 as the second row, finally, we get a $(32 \times 32 \times 1)$ 2D matrix. When the size of the array is not an integer multiple of 32, we pad the array with repeat points. So, we get three 2D matrices which respectively store x, y, z coordinates of points. Similarly, we sort raw data again by x, y value and get six more 2D matrices. Since the coordinate data of the point cloud has not changed after transformation, these 2D matrices still store 3D geometric information. Finally, there are nine 2D matrices concatenated to apply 2D convolutions. Since the size of each point cloud frame is different, we pad those who have smaller sizes with repeat points to make the sizes consistent.

B. Feature Extraction

Different from the data of objects scanned by the 3D scanner, the curvature features of the point cloud scanned by vehicle-mounted LiDAR are more significant. It means the model can pay more attention to the curvature features of point clouds to ensure the reconstruction precision, especially for the contour of point clouds with circular arc distribution. Besides, the calculated curvature can also provide additional information for the encoding process. The curvature feature introduces more information in geometry instead of local information in point cloud sequence, which enhances the overall understanding of the network on point clouds.

We also bring in density information to describe the distribution of point clouds within the outline. The input of the density calculation function is the same as that of the curvature calculation function, which is a point set G of size $(n \times 3)$. The output is the relative density of each point in point set D, whose size is also $(n \times 1)$. Since the global density calculation of each point is costly, we choose a relatively simple way to estimate the partial density. Firstly, for each point, we find k adjacent points of it and calculate the distances from each adjacent point to the selected point: $(x_s - x_i)^2 + (y_s - y_i)^2 + (z_s - z_i)^2 = r_i^2$. Then we calculate the average of these distances: $r_a = \frac{\sum_{i=1}^{k} r_i}{k}$. Finally, the estimated density of selected point D_s can be solved as: $D_s = \frac{k}{\pi r_a^2}$.

C. ConvLSTMlm

ConvLSTM has significant advancements in some applications, but it may not have long memory from a time

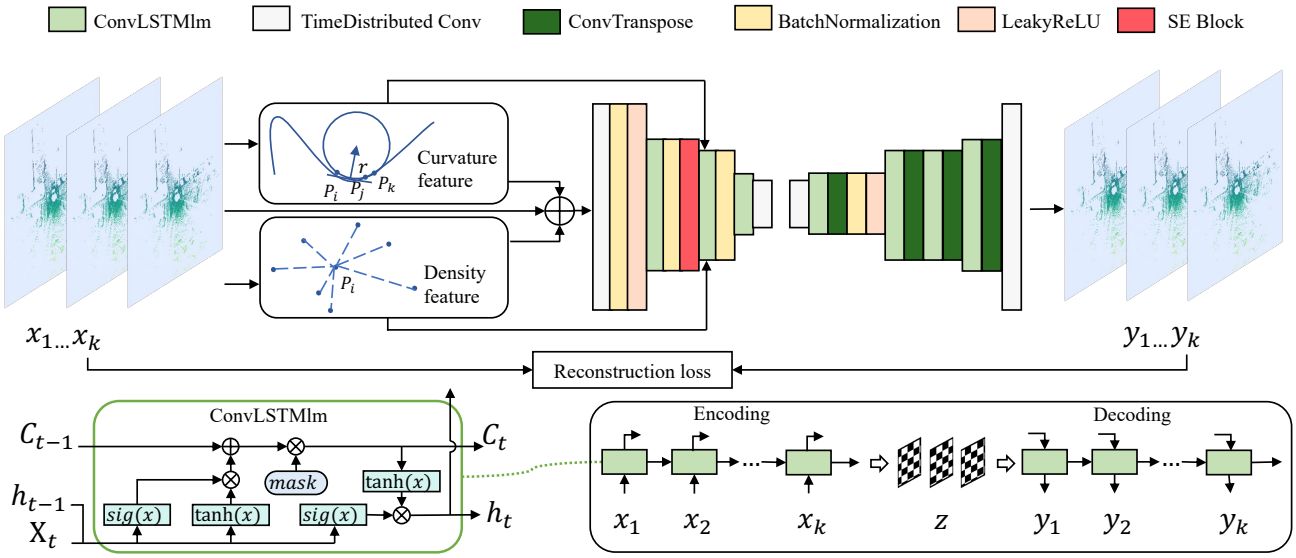

Fig. 1. SPCCNet architectures. The concatenation of raw point cloud stream, curvature, and density feature information is fed to the encoder module as input. The encoder mainly consists of ConvLSTMlm layers and a SE-Block is embedded in it. The decoder also consists of several ConvLSTMlm with the difference being the use of ConvTranspose.

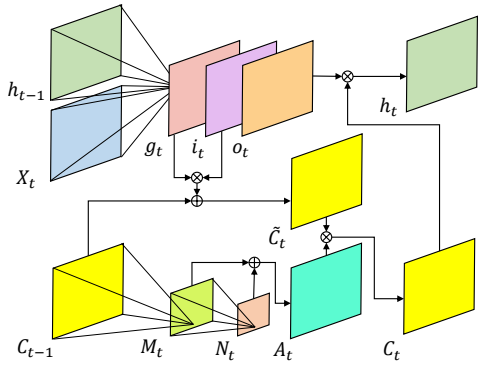

Fig. 2. The ConvLSTMlm layer. The features X_t are input to the ConvLSTMlm layer with h_{t-1} and C_{t-1} of the previous frame. The cell state is enhanced by two layers of convolution.

series perspective which limits it to further improve performance [16]. A natural strategy is to avoid the vanishing gradient problem on the long-term information. We modify the ConvLSTM to enhance the long memory, named ConvLSTMlm shown in Fig. 2. The C_{t-1} is calculated by one layer of convolution to obtain M_t, and M_t is calculated by convolution again to obtain N_t. M_t and N_t are uppooled to the same size as C_{t-1} and then added to get the mask A_t. The mask A_t is multiplied by $\tilde{C}_t$ in the element level to obtain the long-term information C_t. μ denotes unpooling and $ReLU$ denotes rectified linear activation function. The ConvLSTMlm does not increase the number of weight matrices, and the parameters to be trained do not increase much as well. The long-term information and hidden information of the previous point cloud frame are also input to the ConvLSTMlm that processes the current point cloud frame, and the ConvLSTMlm learns the information at different times to better capture the temporal features.

D. Network Architectures

As shown in Fig. 1, we design an asymmetric encoder-decoder structure for efficient streaming point cloud compression. The input is enhanced point cloud information consisting of raw point cloud, curvature, and density. The network compresses and encodes the point cloud blocks through several downsampling layers. ConvLSTMlm is chosen as the main structure for its strong learning ability on long time series data. Then the SE Block (Curvatur-Density-Weight Block) is embedded into the encoder, which can help the encoder focus on more effective feature maps in the compression process. Finally, we use the upsampling layers to decode the encoded features.

At the beginning of the encoder, we use 1 $\times$ 1 2D convolutions to ascend the dimension of data. Then we use 3 $\times$ 3 convolutions in each ConvLSTMlm layer to maintain the ability of encoding. Besides, we set the stride of convolution in each ConvLSTMlm in the encoder as 2 to downsample the data. The stride of the last 2D convolution is also set to 2. The time step of each ConvLSTMlm is set to 5. During the downsampling, spatial and temporal information will be kept in the ConvLSTMlm layers of the encoder. At the same time, the ConvLSTMlm layers of the encoder use historical point cloud information to assist the encoding of the current point cloud.

To better recover point clouds from encoded features, the decoder consists of an alternate combination of modules for decoding and upsampling. As in the encoder, the ConvLSTMlm layers are also deployed. Each ConvLSTMlm layer can provide more precise information for following upsampling layers. We deploy a set of transposed convolution (ConvTranspose) [15] layers, which seek to generate the

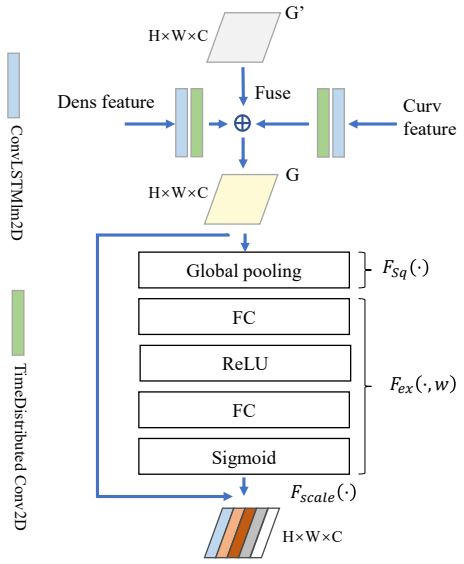

Fig. 3. The feature fusion block in SPCCNet. The curvature and density features are fused into point cloud feature, and an attentive mechanism based on SE Block [10] optimizes the weights of fused features.

input signal by a sum over convolutions of the feature maps (as opposed to the input) with learned filters, instead of simple interpolation or unpooling layers. They provide more trainable parameters to the decoder, which means higher generalization ability.

As shown in Fig. 3, we introduce a SE Block in the encoder to further optimize the weights of fused features. G' is the output of the previous ConvLSTMlm. We fuse the output G' with density and curvature features and get fused output G. F_{sq} performs feature compression according to the spatial dimension, turning each two-dimensional feature channel into a real number. F_{ex} uses the parameter w to generate weights for each feature channel. The output weight is regarded as the importance of each feature channel after feature selection, and the previous features are weighted to complete the re-calibration of the original feature in the channel dimension by F_{scale}.

E. Loss

To learn the overall contour information, based on the node-to-node correspondence, a loss function for training is proposed, which can enable the network to converge faster and fit better. This function consists of two portions: the first portion ℓ_{os} is for improving the degree of overall similarity between raw point clouds and recovered point clouds, which takes the logarithm of the reciprocal of MSE to make it more sensitive to the change of large MSE error than that of small error; the second one (Mean Absolute Error) ℓ_{mae} is used to ensure the precision of point cloud details. As shown below:

$$\min\ \ell = \omega_1 \ell_{os} + \omega_2 \ell_{mae} + b \tag{1}$$

where

$$\ell_{os} = log_{10} \frac{1}{\frac{1}{n}\sum_{i=1}^{n}(\widehat{y}_i - y_i)^2}, and\ \ell_{mae} = \frac{1}{n}\sum_{i=1}^{n}|\widehat{y}_i - y_i| \tag{2}$$

where $\widehat{y}_i$ and y_i represent estimated data and raw data, n is the size of them. The weights ω_1, ω_2 of two portions, and bias b are adjusted according to the epochs of training and the properties of specific datasets. In the early epochs of training, the ℓ_{os} is more significant for shaping the outline, while the ℓ_{mae} is less effective. After obtaining a clear overall outline, we gradually increase ω_2 to obtain more details information.

IV. Experiment

A. Dataset and Implementation Details

In experiments, we use two open-source datasets platform (Robosense [1] and Tier IV [2]). **1)** Robosense has 5000+ frames of point cloud data collected from cities. It consists of 3031 frames of urban main roads, 1359 frames of Grade 2 general urban roads, and 689 frames of intersections. **2)** Tier IV has various datasets from different sources, such as Toyota, Nagoya University, etc. Most of them are collected from cities and villages with connected highways. We select several scenes of Kasugai and Nagoya, which have 24468 frames of point clouds in all.

Our method is implemented on a Tesla V100. The network is implemented by Tensorflow and Keras. For training the network structure, we adopt Adam optimizer with a beta factor of 0.9 and set the learning rate to 0.0003. The batch size is set to be 16 in all experiments. Each point cloud is normalized to [0, 1] and we split 20 % of all the datasets above for testing and the rest for training. In this experiment, our testing datasets mainly contain four representative scenes of campus, city, highway, and village. Besides, we compare the point cloud compression results of Google's Draco [12], G-PCC [4], Octree-based [17], JPEG-based method [18], and our SPCCNet method. Three acknowledged evaluation metrics, i.e., Root Mean Squared Error (RMSE) [19], Structural Similarity (SSIM) [20], and PSNR [11] are selected to evaluate our SPCCNet method.

B. Offline Evaluation

As shown in Fig. 4, in selected Bpps, we visualize the compression results in four scenarios. We can observe that the results of our SPCCNet method can achieve high similarity to the raw point cloud in visual effects in different scenarios. Google's Draco and G-PCC have a similar performance to ours, but the Octree-based method performs a little bit worse. The performance of the JPEG-based method is more unsatisfactory. Furthermore, we can see that from the details in the upper left corner of each plot in Fig. 4, our SPCCNet method shows better continuity and smoothness while the results of other methods are jagged.

Detailed performance information in different scenarios is shown in Table. I. Firstly, with a high compression ratio

[1]https://robosense.ai/resource
[2]https://rosbag.tier4.jp

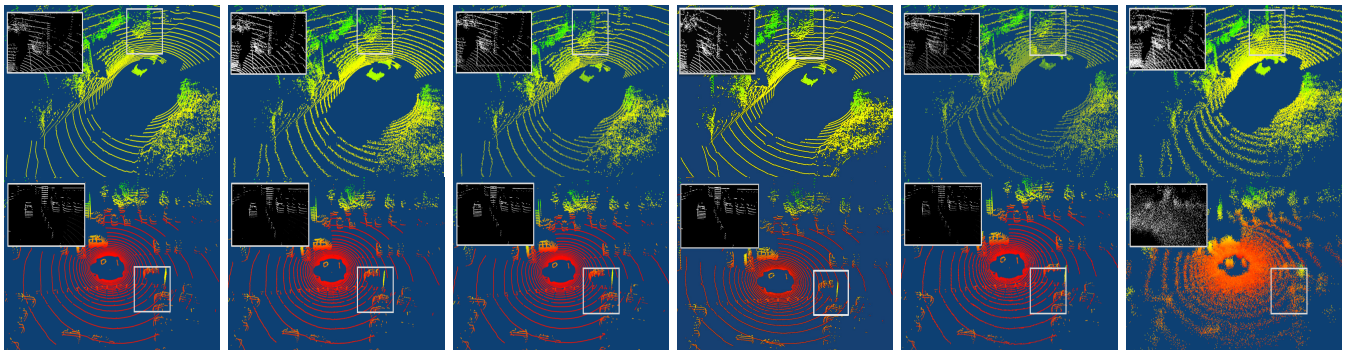

Fig. 4. The comparison of quality performance between ours and other methods in different scenarios, in which campus and highway are in order from up to down. Each column corresponds to different methods, including ground truth, our SPCCNet method, Google's Draco, G-PCC, Octree-based, and JPEG-based method from left to right. Note that these experimental results are under different Bpps, specifically our method uses the minimum Bpp which is 1.1, while other methods correspond to the number in the second row of each method in Table. I separately.

TABLE I

THE COMPRESSION PERFORMANCE OF DIFFERENT METHODS.

Methods \ Scenarios	Campus				City				Highway				Village			
	Bpp	RMSE	SSIM	PSNR	Bpp	RMSE	SSIM	PSNR	Bpp	RMSE	SSIM	PSNR	Bpp	RMSE	SSIM	PSNR
Our SPCCNet	1.1	0.015	0.829	36.214	1.1	0.047	0.835	38.721	1.1	0.048	0.799	26.332	1.1	0.007	0.998	33.681
	1.9	0.009	0.866	40.561	1.9	0.038	0.858	40.680	1.9	0.048	0.799	26.452	1.9	0.006	0.999	38.167
	4.1	0.005	0.892	45.990	4.1	0.031	0.891	42.765	4.1	0.047	0.799	26.491	4.1	0.006	0.999	40.432
Draco	5.3	0.057	0.899	24.819	5.7	0.042	0.912	27.466	4.1	0.078	0.786	22.773	6.2	0.064	0.884	23.856
	5.6	0.735	0.900	24.948	5.9	0.039	0.915	28.156	4.9	0.075	0.787	22.799	6.6	0.059	0.888	24.516
	21.4	0.146	0.978	32.880	21.3	0.021	0.997	35.743	17.5	0.060	0.788	23.552	22.5	0.001	0.999	46.298
G-PCC	2.2	0.065	0.758	23.768	1.1	0.071	0.756	23.232	1.9	0.162	0.726	17.143	2.7	0.071	0.717	23.040
	6.0	0.051	0.786	25.809	1.8	0.054	0.772	25.216	4.2	0.095	0.763	22.494	6.5	0.061	0.737	24.297
	23.5	0.019	0.950	34.033	2.3	0.053	0.775	25.407	12.9	0.055	0.785	25.501	18.8	0.037	0.875	28.566
Octree-based	4.2	0.114	0.865	18.842	4.3	0.132	0.818	17.592	3.9	0.130	0.763	17.737	3.2	0.197	0.870	14.122
	4.7	0.099	0.878	20.124	4.6	0.116	0.866	18.682	4.7	0.117	0.768	18.642	4.2	0.103	0.872	19.780
	15.7	0.041	0.899	27.863	15.7	0.061	0.893	24.332	15.7	0.087	0.772	21.258	15.7	0.059	0.888	24.516
JPEG-based	4.0	0.147	0.029	16.680	4.0	0.181	0.015	14.818	4.0	0.178	0.002	14.982	4.0	0.197	0.027	14.098
	4.8	0.100	0.030	19.999	4.8	0.133	0.017	17.555	4.8	0.153	0.002	16.331	4.8	0.131	0.027	17.677
	5.2	0.058	0.038	24.763	5.2	0.078	0.022	22.203	5.2	0.133	0.002	17.517	5.2	0.099	0.031	20.004

(low Bpp), our SPCCNet method is obviously superior to other methods, especially for PSNR and RMSE, which are highlighted in Table. I. In the campus scenario, the PSNR of our model can reach 45.9, and in the village scenario, the RMSE is only 0.006. This is due to our SPCCNet based on ConvLSTMlm effectively learning the spatial and temporal characteristics of point cloud data. Google's method also shows satisfactory performance, followed by G-PCC method. Secondly, compared with other methods, the advantages of our SPCCNet method are not obvious in SSIM. On the contrary, Google's Draco and Octree pay more attention to the internal structure of the point cloud. So they show better performance in SSIM rather than in PSNR. Finally, compared with other scenarios, the compression performance of the highway scenario is worse. The reason is that the point clouds in the highway scenario are sparser, so it is difficult to extract effective feature data.

C. Ablation Study

In Table. II, we study the performances of networks with different features and use the same metrics mentioned above to evaluate the results in experiments: 1) MSE-w/o-curv: train network under MSE loss without curvature introduced. 2) MSE-w/o-dens: train network under MSE loss without density. 3) ℓ-w/o-curv: train network under proposed loss without curvature. 4) ℓ-w/o-dens: train network under proposed loss without density. 5) MSE: train network under MSE with both curvature and density. 6) ℓ_{os}: train network under only the first part of the proposed loss. 7) ℓ-w/o-SE:

TABLE II

ABLATION STUDIES ON TIER IV DATASET.

	RMSE$^-$	SSIM$^+$	PSNR$^+$	Time/frame (ms)
MSE-w/o-curv	0.028	0.809	31.123	138.4
MSE-w/o-dens	0.022	0.815	33.245	143.2
ℓ-w/o-curv	0.017	0.847	35.332	138.4
ℓ-w/o-dens	0.014	0.855	37.156	143.2
MSE	0.021	0.831	33.730	163.2
ℓ_{os}	0.010	0.863	40.114	163.2
ℓ-w/o-SE	0.009	0.859	40.571	132.8
ConvLSTM	0.010	0.849	39.742	163.1
Ours	**0.008**	**0.872**	**42.231**	163.0

TABLE III
IOU SCORES ON SEQUEEZESEGV2 OF DIFFERENT CLASSES.

	Bpp$^-$	Car	Pedestrian	Bicyclist	Average
Raw point cloud	\	84.2	18.4	64.9	32
Draco	8.2	79.7	16.6	63.2	29.3
SPCCNet	1.1	82.8	17.6	63.8	30.4

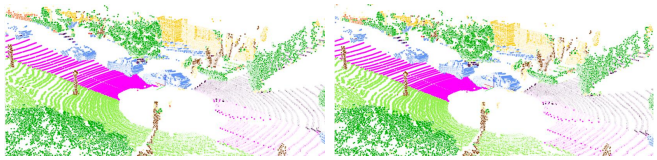

Fig. 5. Semantic segmentation performance of raw point cloud (left) and our method (right).

train network without SE Block under proposed loss.

Our proposed SPCCNet achieves the best performances of three metrics, while the others without curvature or density reach lower. Besides, the lack of SE Block also weakens the performance of the network, which indicates that SE Block does extract feature maps that are effective for compression. Finally, by removing the forget gate and performing a two-layer convolution on the long-term information to form a mask, the model obtains better quality performance.

D. Application

To further demonstrate the effectiveness, we evaluate our SPCCNet method on Semantic segmentation. We use SqueezeSegV2 [21] for the semantic segmentation task. We apply our method and Google's Draco on the raw point cloud and get the decoded point cloud data for the semantic segmentation task. We set the Bpps of our SPCCNet and Draco at 1.1 and 8.2 respectively. Intersection-over-union (IOU) [22] is used to evaluate the performance of the semantic segmentation task. The semantic segmentation result is shown in Fig. 5, and we can observe that compared to the raw point cloud, our method can keep fine-grained details at a relatively low Bpp. More details are shown in Table. III. Our method achieves higher IOU than Draco on all classes and is closer to the result of the raw point cloud, which means we can ensure the performance on the semantic segmentation task while maintaining a high compression rate.

V. CONCLUSION

This paper proposes an efficient learning-based end-to-end streaming point cloud compression network named SPCCNet. Specifically, the curvature and density information are introduced to help our SPCCNet learn the spatial features of point clouds effectively. Moreover, we propose ConvLSTMlm to capture the temporal features of point cloud sequences, which removes the forget gate and performs a two-layer convolution on the long-term information. The experimental results on Robosense and Tier IV show that our SPCCNet can compress point cloud data with a higher compression ratio while maintaining the quality of the data. We believe that our compression can be applied to V2X scenarios and downstream perception tasks without hindering performance.

REFERENCES

[1] Y. Hu, Y. Lu, R. Xu, W. Xie, S. Chen, and Y. Wang, "Collaboration helps camera overtake lidar in 3d detection," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2023, pp. 9243–9252.

[2] C.-S. Liu, J.-F. Yeh, H. Hsu, H.-T. Su, M.-S. Lee, and W. H. Hsu, "Bird-pcc: Bi-directional range image-based deep lidar point cloud compression," in *ICASSP IEEE Int Conf Acoust Speech Signal Process Proc*. IEEE, 2023, pp. 1–5.

[3] T. Beemelmanns, Y. Tao, B. Lampe, L. Reiher, R. van Kempen, T. Woopen, and L. Eckstein, "3d point cloud compression with recurrent neural network and image compression methods," in *IEEE Intell Veh Symp Proc*. IEEE, 2022, pp. 345–351.

[4] S. Schwarz, M. Preda, V. Baroncini, M. Budagavi, P. Cesar, P. A. Chou, R. A. Cohen, M. Krivokuća, S. Lasserre, Z. Li, J. Llach, K. Mammou, R. Mekuria, O. Nakagami, E. Siahaan, A. Tabatabai, A. M. Tourapis, and V. Zakharchenko, "Emerging mpeg standards for point cloud compression," *IEEE Jour. Emer. Select. Top. Circu. Syste.*, vol. 9, no. 1, pp. 133–148, 2019.

[5] J. Zhang, T. Chen, D. Ding, and Z. Ma, "Yoga: Yet another geometry-based point cloud compressor," in *MM - Proc. ACM Int. Conf. Multimed.*, 2023, pp. 9070–9081.

[6] W. Yan, S. Liu, T. H. Li, Z. Li, G. Li *et al.*, "Deep autoencoder-based lossy geometry compression for point clouds," *arXiv*, pp. arXiv–1905, 2019.

[7] D. T. Nguyen, M. Quach, G. Valenzise, and P. Duhamel, "Lossless coding of point cloud geometry using a deep generative model," *IEEE Trans. Circuits Syst. Video Technol.*, 2021.

[8] C. Fu, G. Li, R. Song, W. Gao, and S. Liu, "Octattention: Octree-based large-scale contexts model for point cloud compression," in *Proc. AAAI Conf. Artif. Intell.*, vol. 36, no. 1, 2022, pp. 625–633.

[9] R. Song, C. Fu, S. Liu, and G. Li, "Efficient hierarchical entropy model for learned point cloud compression," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2023, pp. 14 368–14 377.

[10] J. Hu, L. Shen, and G. Sun, "Squeeze-and-excitation networks," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, June 2018.

[11] Q. Huynh-Thu and M. Ghanbari, "Scope of validity of psnr in image/video quality assessment," *Electron. Lett.*, vol. 44, no. 13, pp. 800–801, 2008.

[12] Google, "Draco 3d," https://google.github.io/draco\#, 2017.

[13] J. Wang, H. Zhu, H. Liu, and Z. Ma, "Lossy point cloud geometry compression via end-to-end learning," *IEEE Trans. Circuits Syst. Video Technol.*, pp. 1–1, 2021.

[14] Y. Zhang, Q. Hu, G. Xu, Y. Ma, J. Wan, and Y. Guo, "Not all points are equal: Learning highly efficient point-based detectors for 3d lidar point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recog.*, 2022, pp. 18 953–18 962.

[15] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Proc. IEEE Comput Soc Conf Comput Vision Pattern Recognit*. IEEE, 2010, pp. 2528–2535.

[16] J. Zhao, F. Huang, J. Lv, Y. Duan, Z. Qin, G. Li, and G. Tian, "Do rnn and lstm have long memory?" in *Int. Conf. Machin. Learn., ICML*. PMLR, 2020, pp. 11 365–11 375.

[17] P. C. Library, "Octree," https://pointclouds.org/, 2020.

[18] P. C. D. Rufael Mekuria, Kees Blom, "Design, implementation, and evaluation of a point cloud codec for tele-immersive video," *IEEE Trans. Cir. and Sys. for Video Technol*, vol. 27, no. 4, pp. 828–842, 2017.

[19] A. G. Barnston, "Correspondence among the correlation, rmse, and heidke forecast verification measures; refinement of the heidke score," *Weather Forecast.*, vol. 7, no. 4, pp. 699–709, 1992.

[20] A. Hore and D. Ziou, "Image quality metrics: Psnr vs. ssim," in *Proc. Int. Conf. Pattern Recognit.* IEEE, 2010, pp. 2366–2369.

[21] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," in *Proc IEEE Int Conf Rob Autom*. IEEE, 2019, pp. 4376–4382.

[22] Z. Shi, S. He, J. Sun, T. Chen, J. Chen, and H. Dong, "An efficient multi-task network for pedestrian intrusion detection," *IEEE T. Intell. Veh.*, pp. 1–1, 2022.