IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received February 22, 2021, accepted March 10, 2021, date of publication March 30, 2021, date of current version April 12, 2021.

Digital Object Identifier 10.1109/ACCESS.2021.3069896

Segmentation of Points in the Future: Joint Segmentation and Prediction of a Point Cloud

CHENG WENCAN[1], (Graduate Student Member, IEEE), AND JONG HWAN KO[2], (Member, IEEE)

[1]Department of Artificial Intelligence, Sungkyunkwan University, Suwon 16419, South Korea
[2]College of Information and Communication Engineering, Sungkyunkwan University, Suwon 16419, South Korea

Corresponding author: Jong Hwan Ko (jhko@skku.edu)

This work was supported by the National Research Foundation grant (NRF 2019R1F1A1048115) and the Institute of Information and Communication Technology Planning & Evaluation (IITP) grants on the AI Graduate School program (IITP-2019-0-00421), ICT Creative Consilience program (IITP-2020-0-01821), and AI Industry Technology R&D program (IITP-2021-0-00066) funded by the MSIT(Ministry of Science and ICT) of the Korea government.

ABSTRACT Recognizing and predicting future three-dimensional (3D) scenes are crucial steps for real-time vision-based control systems, as these steps enable them to react appropriately in advance. In this study, a method for predicting the position of a 3D point cloud in the future and simultaneously segmenting the predicted point cloud is proposed for the first time. The prediction and segmentation tasks are performed by a novel neural network architecture that extracts both local geometric features and flow features for joint segmentation and prediction. Furthermore, we propose a new evaluation metric for future point cloud segmentation to resolve the problem of inconsistency in the order of future point clouds. The results of experiments conducted using real-world large-scale benchmark datasets revealed that the proposed network achieves higher prediction and segmentation accuracy than other baseline methods.

INDEX TERMS Point cloud, recurrent neural network, 3D reconstruction, segmentation.

I. INTRODUCTION

As three-dimensional (3D) vision data can provide abundant spatial information, it is being widely used in many areas, including autonomous driving and mobile robots [1]. With recent advances in LiDAR technology that enable quick collection of 3D point clouds with high density and accuracy [2], recognition of point clouds has become a significant research topic. However, direct processing of point clouds has been a big challenge because the point cloud acquired by LiDAR is irregularly sampled, unstructured, and unordered in general. Since the introduction of PointNet [3], which was designed to directly extract unordered point cloud features by a symmetric structure, many researchers have proposed PointNet-based deep neural network models that directly utilize point clouds for various tasks including classification, segmentation, and detection. However, all of the existing studies focused on static point clouds in the past/current time frame [4]–[7].

The associate editor coordinating the review of this manuscript and approving it for publication was M. Venkateshkumar.

In applications such as autonomous driving, it is crucial that the vehicle quickly responses to changes in the external environment. In order to respond to future events in advance, systems should be equipped with the ability to predict and understand future scenes. Therefore, joint prediction and segmentation of the future 3D point cloud is a critical task in such practical applications. Recently, Fan *et al.* proposed PointRNN [8], which predicts point clouds in the future. However, although it can perform future scene prediction, it does not have the ability to segment the predicted scenes. Scene point cloud sequence forecasting [9] has also been proposed to reconstruct and predict object trajectories using point clouds; however, its trajectory prediction is limited to a road agent, and the trajectories of other important objects such as roads and obstacles are not predicted. To provide a better understanding of future 3D scenes, this paper introduces a method that accomplishes joint segmentation and prediction of future point clouds by utilizing a novel DNN model that can simultaneously predict future point clouds and perform semantic segmentation of the predicted point clouds. The proposed model aims not only to track the movement of

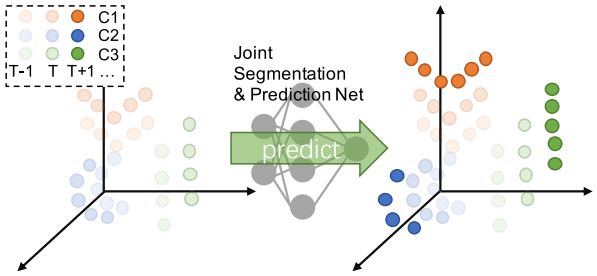

FIGURE 1. Joint segmentation and prediction of point cloud. Our method directly outputs the point-wise segmentation labels (C1, C2, C3,...) as well as predicted coordinates of the future point cloud(T+1,...), based on the past point cloud(..., T−1, T).

each point in the sequential point cloud, but also to extract crucial semantic information.

To achieve this goal, the proposed model is equipped with both hierarchical geometric feature extraction and flow extraction abilities for segmentation and prediction, respectively. In addition, it employs a recurrent neural network (RNN) structure to record the extracted geometric and flow features from the previous input frames, and predict the corresponding geometric and flow features for the upcoming frames. Based on the predicted geometric features and flow features, the proposed model can predict point-wise relative displacement and semantic segmentation labels. Additionally, the proposed model follows the PointNet-based symmetric design [3], [7] so that it can handle an input point cloud with irregular order.

Similar to FlowNet3D [10], the flow extraction function substitutes the embedding flow between two continuous frames into an RNN to construct the point-wise relative displacement. The hierarchical geometric feature extraction structure implemented in the proposed model is introduced by PointNet++ [7], which extracts the global feature of the point cloud through a symmetric operation after obtaining the local features of different sampling levels. After extracting hierarchical geometric features and flow features from different past continuous frames, the proposed model embeds these features as time-based sequences and predicts corresponding feature embeddings for the upcoming frames by using an RNN unit. Then, the feature embeddings are used for prediction and segmentation by the feature propagation operations [10].

We used real LIDAR scan datasets, SemanticKITTI [12], and Argoverse [13] to train and evaluate our proposed model. The experimental results revealed that our model performs segmentation and prediction with higher accuracy than other baseline methods. The key contributions of this paper are as follows:

- To the best of our knowledge, this is the first time a task of joint segmentation and prediction of the future point cloud has been introduced.
- The paper proposes the first complete neural network architecture for joint segmentation and prediction of point clouds by introducing two segmentation and prediction constraints.
- We introduce a novel training method for the joint prediction and segmentation task, and show that our trained network can achieve a better performance than other methods on real-world LIDAR benchmarks.

II. RELATED WORK

A. POINT CLOUD SEMANTIC SEGMENTATION (PCSS)

PCSS generates semantic labels for each point [14] through supervised learning structures. As the raw point cloud is difficult to deal with directly because of its disordered characteristics, several existing methods convert point clouds into different representations such as multi-view [15], voxel [16], [17], or range images [18]–[20] to regularize and discretize point clouds. Then, the regularized representations can be processed using the conventional convolutional neural network for semantic segmentation.

However, these methods incur large computational overhead owing to the introduction of redundancy in the regularization process, and they also cause the resolution to be lost owing to the discretization operation. Qi *et al.* proposed a concise and effective semantic segmentation approach that directly processes raw points using PointNet [3]. PointNet uses a symmetric operation to extract bottleneck global features from the disordered point cloud, which are then used to perform classification or segmentation tasks. Since the introduction of PointNet, researchers have proposed several other methods that outperform PointNet on various benchmark datasets. One category of studies [4]–[7] focused on optimizing local features to improve segmentation performance. There are also approaches that use the recurrent network structure to obtain more effective global features. Meanwhile, recent studies have indicated that the graph network architecture can make good use of the geometric topology of the point cloud to extract dynamic features and enrich the representation power of the point cloud. It should be noted that the above direct PCSS methods are all based on the retained bottleneck framework, that is, they have to generate global features [21]–[23]. Although the proposed methods [24], [25] also inherit this structure to complete segmentation, the difference is that the proposed method generates a global feature of the predicted frame instead of the current frame.

B. POINT CLOUD GENERATION

A point cloud generation task generates a point cloud from a given set of features that represent a point cloud in the high-dimensional latent space. As for the prediction case, the task generates a point cloud of future scenes from features extracted from past frames. As mentioned in the PCSS subsection, extraction of the bottleneck features relies on a symmetric operation, which makes determining the order of the generated point cloud difficult. Although a fully

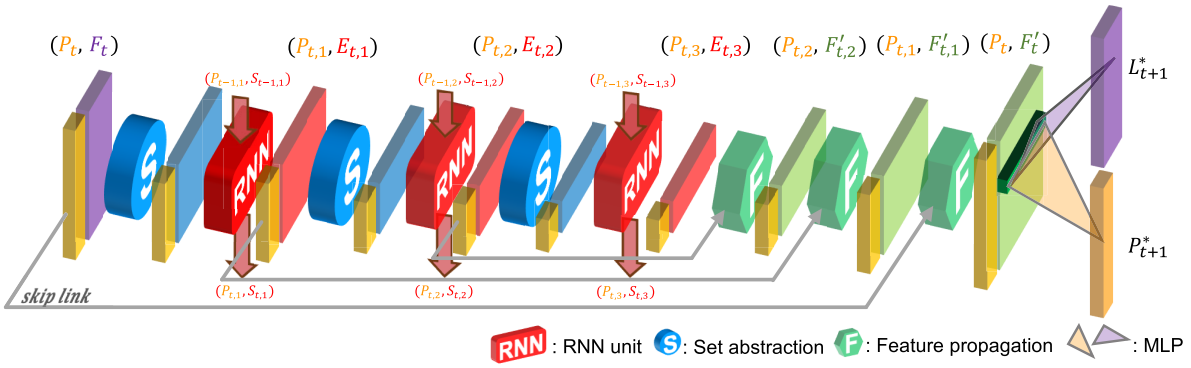

FIGURE 2. Overall architecture of joint segmentation and prediction. It is based on a hierarchical seq2seq architecture that contains a stack of set abstraction layers and RNN units for multi-scale feature extraction, and a stack of feature propagation layers for propagation from high-level features to point-wise features. At the end of the architecture, two MLP branch are employed for segmentation and prediction.

connected layer has the ability to rebuild a specific order [26]–[29], the network built by this method requires a considerable matrix to store weights. To solve these problems, some studies [28], [30] have proposed methods that obtain the order of points in the reconstructed point cloud through a lightweight 2-dimensional plane. PointAE [31] introduces bias into the network so that the network can activate bias through the bottleneck feature to determine the output order of the point cloud. In other studies [8], [10], the order of the input point cloud has been maintained through a skip link. However, these methods are applicable when the input point cloud is given. Moreover, the features generated from PointRNN [8] are only able to establish flow estimation and not segmentation. To resolve this problem, we designed a joint segmentation and prediction network that can extract not only the flow features but also the geometric features for segmentation.

III. PROBLEM STATEMENT

This paper introduces a new joint segmentation and prediction task, which predicts the future sequence and the corresponding label of each point in a point cloud. $P_{\Delta t}$ and $L_{\Delta t}$ are obtained based on the input point cloud sequence P_t and the features F_t of the point cloud, where $t \in [-T+1, \ldots 0]$ denotes the frame timestamp t in the past and $\Delta t \in [1, \ldots \Delta T]$ denotes the frame timestamp in the future. $P_t = \{p_t^i\}_{i=1}^N$ denotes an N-length point set of a specific timestamp with $p_t^i \in R^d$ and $i = 1, \ldots N$. Here, we only focus on the Euclidean space such that $d = 3$. $F_t = \left\{f_t^j\right\}_{j=1}^N$ denotes the input feature set with the same length and order as the input point set P_t and $f_t^j \in R^k$, where k is the dimension of the feature vector. F_t can be an additional RGB color or the LIDAR intensity. In this study, we use F_t as a point-wise semantic label of the corresponding input point set. Finally, $L_{\Delta t} = \left\{l_{\Delta t}^j\right\}_{j=1}^N$ denotes the predicted future semantic labels with the same order and size as the reconstructed feature points $P_{\Delta t}$. Therefore, our goal was to learn a function $f : \{P_t\, X_t\}_{t=1-T}^0 \longrightarrow \{P_{\Delta t}\, L_{\Delta t}\}_{\Delta t=1}^{\Delta T}$ that maps the past points and features to the points and semantic labels in the future frames.

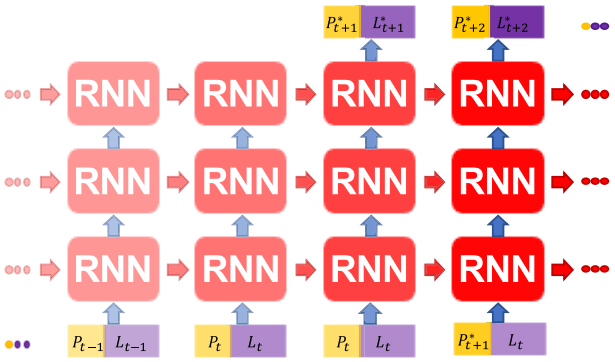

FIGURE 3. Temporal Seq2Seq architecture. The bottom RNN units require point sets and their corresponding labels. The upper RNN units receive the output from the lower RNN units as input to generate more global features (blue flow). As the time goes (from left to right), each RNN unit leverages their states from last timestamp to update current states (red flow).

IV. JOINT SEGMENTATION AND PREDICTION NETWORK

A. ARCHITECTURE

For joint prediction and segmentation, we introduced a hierarchical architecture with several various components, as shown in Fig. 2. First, for each timestamp, a set abstraction (SA) layer is applied for down-sampling and local feature extraction, and the RNN unit is utilized for fusing the current feature with previous features and encoding a single local feature. Alternate stacking of these two units generates global features. Then, to infer point-wise features, the subsequent feature propagation (FP) layers combine and propagate the features with different scales in each layer of the feature extraction stage. Finally, two independent shared-weight multi-layer perceptron (MLP) branches [3] are adopted on the point-wise features, which are used for prediction and segmentation for the next timestamp. Figure 3 shows the proposed architecture expanded in a temporal sequence.

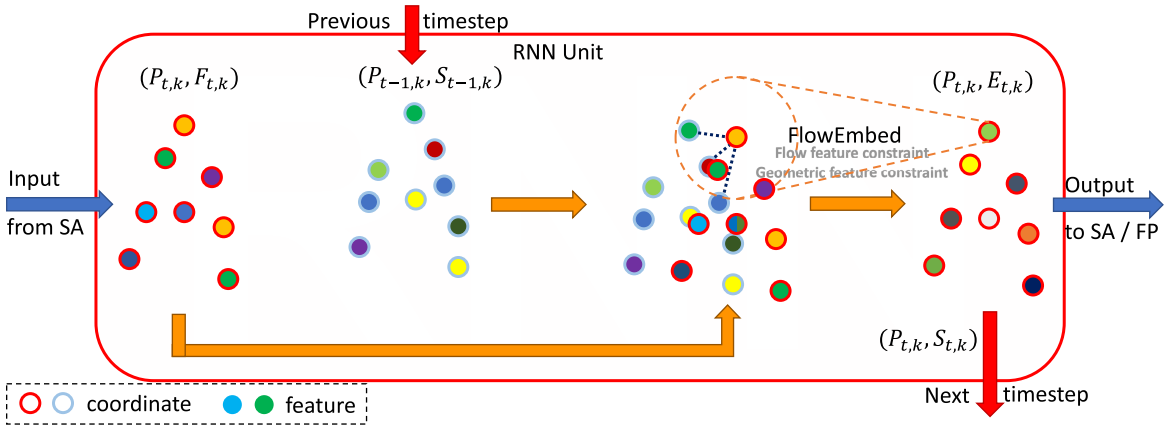

FIGURE 4. Architecture of the RNN Unit. The RNN unit requires current coordinates and corresponding labels as the input and adopt flow embedding layer with two constraints(flow feature constraint & geometric feature constraint) to extract and fuse flow feature and local geometric feature and update its state.

B. RNN UNIT

As the model needs to extract flow and local geometric features for prediction and segmentation, we use an RNN unit with flow embedding (FE) layers [7] as shown in Fig.4. The RNN unit extracts features from the neighboring regions in previous frames around centers formed by current input points. Therefore, our model can obtain the relative geometric and semantic information that benefits the joint task of prediction and segmentation.

Other existing models, such as FlowNet3D [10] and PointRNN [8], also use the FE layers for flow voting. However, we found that this layer can also be effectively used to vote for local geometric features for segmentation. Based on this observation, we used the FE layer to collect points and features from the previous frame and vote for the flow and local geometric information based on these collected features. Specifically, we activated the FE layers to extract semantic features from the collected points by introducing constraints into the network that FlowNet3D and PointRNN do not provide. The constraint was introduced by adding additional MLP layers at the end of the network, and forcing the output of the last MLP layer to be the segmentation labels. To obtain higher segmentation accuracy, the FE layer needs to achieve better voting for local geometric features as much as possible to match this geometric constraint. Because the FE layer can also complete the extraction of the current and previous frame point flow, the final output becomes a fusion of the flow and local geometric features.

For timestamp t, the current frame data is given by the input $(P_{t,l}, F_{t,l})$, and the the previous frame data is the state $(P_{t-1,l}, S_{t-1,l})$ stored in the RNN unit, where the l indicates the l-th RNN unit. The FE layer determines the center point $p^i_{t,l}$ in $P_{t,l}$ and extracts neighbors $p^j_{t-1,l} \in \mathcal{N}_{P_{t-1,l}}(p^i_{t,l})$ from the $P_{t-1,l}$ point set around it. Naturally, the neighbor set sampled from the previous frame $\{p^j_{t-1,l}, s^j_{t-1,l}\}_{j|p^j_{t-1,l} \in \mathcal{N}_{P_{t-1,l}}(p^i_{t,l})}$ can be used to vote for local geometric features. Meanwhile, the flow features can be generated from the point-wise displacement $p^j_{t-1,l} - p^i_{t,l}$ with respect to the center point $p^i_{t,l}$. Then, fusion of the geometric and flow features is performed by combining the features $f^i_{t,l}$ of the corresponding $p^i_{t,l}$ and state $s^j_{t-1,l}$ carried by $p^j_{t-1,l}$

$$s^i_{t,l} = \underset{p^j_{t,l} \in \mathcal{N}_{P_{t-1,l}}(p^i_{t,l})}{MAX} (h_{\Theta_f}(f^i_{t,l}, s^j_{t-1,l}, p^j_{t-1,l} - p^i_{t,l})), \quad (1)$$

where h is a non-linear function with shared parameters Θ_f, and MAX is the channel-wise max pooling operation. Hence, the final extracted features will include local geometric information and flow information.

C. DOWN-SAMPLING AND UP-SAMPLING

To achieve effective segmentation, local features and global features need to work together in the model. However, as the proposed RNN unit is designed to obtain local information, a broad range of global features is difficult to extract. To better extract the global features, we built a hierarchical structure based on the structure proposed in PointNet++. Because the point cloud is irregular and disordered, PointNet++ uses a combination of SA, which is designed for down-sampling and grouping, and FP, which is designed for up-sampling and feature propagation. With these key layers, PointNet++ performs the extraction of features of various scales, from local (low level) to global (high level) features, and establishes segmentation at the static points.

In PointNet++, the SA layer uses fastest-point sampling (FPS), which samples n regions in $P_{t,l-1}$ from the previous layer to generate the subset $P_{t,l}$ formed by the centers of these regions, where l indicates the l-th SA layer. With each point $p^i_{t,l} \in P_{t,l}$ as the center, the features of the points $\mathcal{N}_{P_{t,l-1}}(p^i_{t,l})$ contained in the surrounding region can be extracted by the

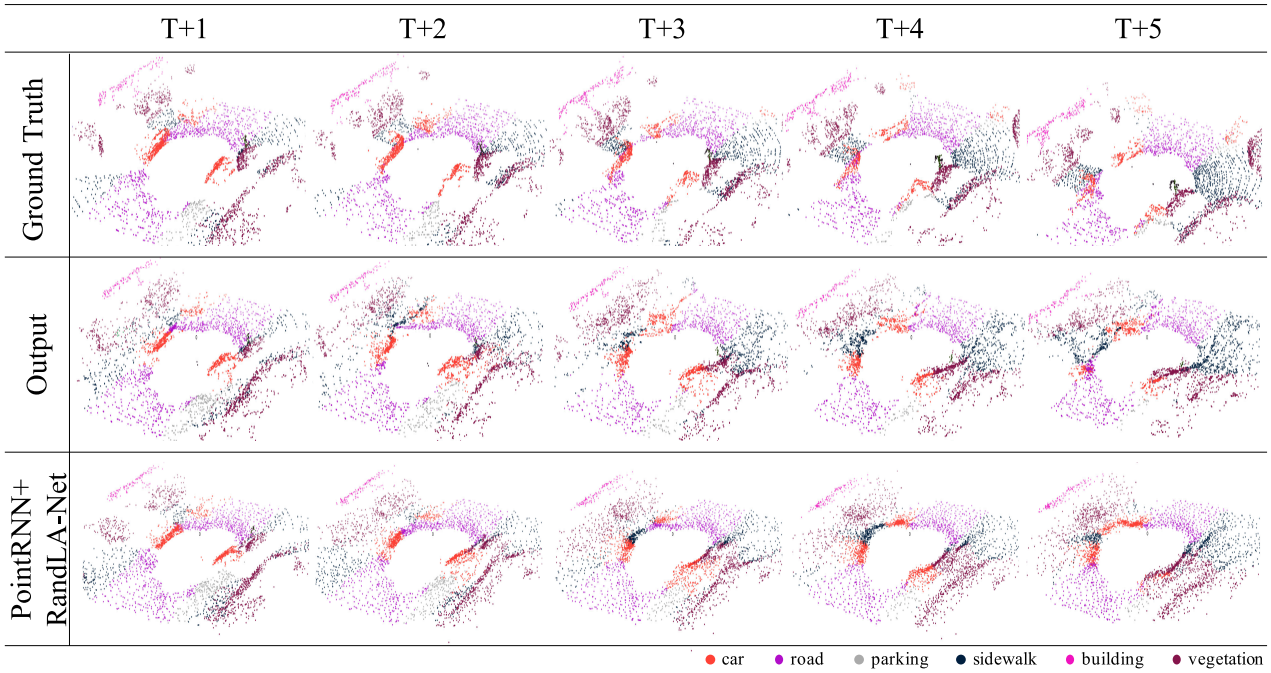

FIGURE 5. Visualization of the segmentation and prediction output on the SemanticKITTI benchmark. The colors indicate the different semantic labels of each point.

SA layer with the following symmetric function:

$$f_{t,l}^{i} = \underset{p_{t,l-1}^{j} \in \mathcal{N}_{P_{t,l-1}}(p_{t,l}^{i})}{MAX} (h_{\Theta_s}(f_{t,l-1}^{j}, p_{t,l-1}^{j} - p_{t,l}^{i})). \quad (2)$$

After l-th level down-sampling is done by the l-th SA layer, the l-th RNN unit fuses the flow and geometric features of the l-th level. These two steps are repeated several times to make the point set continually sparser. The features it packs become more global as a result.

After the multi-scale feature has been extracted, it is necessary to propagate the feature from a high level back to a low level, and infer the point-wise feature for the final segmentation and prediction. PointNet++ uses the FP layer that interpolates features of the high-level points (sparse) at the coordinates of the low-level points (dense). Pointet++ utilizes the FP layer to complete segmentation, and FlowNet3D [10] and PointRNN [8] also implement reconstruction based on the FP layer.

$$f_{t,l}^{i'} = \frac{\sum_{j=1}^{k} w_i(p) f_{t,l+1}^{j}}{\sum_{j=1}^{k} w_i(x)} \quad \text{where} \quad w_i(p) = \frac{1}{||p_{t,l}^{i}, p_{t,l+1}^{j}||^2}. \quad (3)$$

D. TRAINING

In a joint prediction and segmentation task, the order of the predicted point cloud can be different from that of the ground truth point cloud.

To eliminate this influence of the order variance, we define two independent training losses for prediction and segmentation.

1) PREDICTION LOSS

As the prediction of point clouds can be regarded as a reconstruction, we consider commonly used loss functions for the disordered point cloud reconstruction, the chamfer distance (CD) [32] loss and the earth mover's distance (EMD) [33] loss. They all measure the relative distance between two sets of point clouds in an arbitrary order and try to minimize the distance between the points.

The CD loss measures the distance between each point in one point set and its closest target point in the other point set; therefore, its action is independent of the order of the points. Assuming P and P^* as the ground truth and predicted point cloud, respectively, the CD loss between these two point cloud sets is defined as

$$\mathcal{L}_{CD} = \sum_{p \in P} \underset{p^* \in P^*}{min} ||p^* - p||_2^2 + \sum_{p^* \in P^*} \underset{p \in P}{min} ||p^* - p||_2^2. \quad (4)$$

In practice, the CD loss can produce reasonably high-quality results. However, because it focuses on the nearest neighbor, point cloud reconstruction using the CD loss can get stuck in local minima, causing the aggregation phenomenon from a visual perspective. To avoid this problem, we use the EMD loss as a supplement.

The EMD loss solves an optimization problem to obtain the optimal point-wise bijection mapping between a pair of point sets. This mapping guides the computation of distance measuring for two point sets by computing the Euclidean distance of the points at both ends of this mapping. The EMD loss is defined as

$$\mathcal{L}_{EMD} = \underset{\phi: P^* \to P}{min} ||p^* - \phi(p^*)||_2^2, \quad (5)$$

where $\phi : P^* \to P$ is the optimal bijection.

Because the bijection mapping is optimal, it ensures that the mapped point for each point in P^* is unique. Therefore, the order of points is irrelevant for mapping.

2) SEGMENTATION LOSS

As mentioned above, the order of the predicted points may be inconsistent with the order of the ground truth points. Meanwhile, the order of the predicted segmentation labels is bound to the predicted points because of the point-wise calculation of MLP layers. Therefore, the order of the predicted label can be inconsistent with the ground truth label. Consequently, computing the point-wise segmentation loss of the disordered point cloud requires aligning the ground truth and the predicted segmentation labels. To tackle this challenge, we used the relative positional relationship between the predicted and ground truth points. To achieve this alignment, we adopted the k-nearest neighbor(kNN) algorithm that takes the majority vote of the segmentation label for each predicted point among the k-nearest neighbor points from the well-labeled ground truth points around it. Then, we can generate the aligned labels by

$$l^{i\dagger} = \underset{v}{argmax} \sum_{(p^i, l^i) \in (P, L)} I(v = l^i). \tag{6}$$

This operation forces the learning objectives to align the labels and the predicted points in the same order. Then, the loss of the predicted labels can be measured by the following element-wise softmax cross-entropy function, which is commonly used for segmentation tasks.

$$\mathcal{L}_{SEG} = \sum_{j=1}^{N} l^{j\dagger} \log s^{j*} \Big|_{s^{j*} = \frac{e^{l^{j*}}}{\sum_{k=1}^{N} e^{l^{k*}}}}. \tag{7}$$

By combining the aforementioned loss functions, the total loss of our proposed end-to-end prediction and segmentation network is

$$\mathcal{L}(P*, P) = \mathcal{L}_{CD} + \alpha \mathcal{L}_{EMD} + \beta \mathcal{L}_{SEG}, \tag{8}$$

where $\alpha, \beta > 0$ are the hyperparameters that balance the loss terms.

V. EXPERIMENTAL RESULTS

A. EXPERIMENT SETTINGS

We evaluated the performance of the proposed model implemented as in Table 1, using large-scale real LIDAR datasets, SemanticKITTI [12], and Argoverse [13]. We conducted our experiments on an NVIDIA RTX2080Ti GPU with TensorFlow. For training, we used the Adam optimizer [34] with its suggested default parameters of beta1 = 0.9 and and beta2 = 0.999, and a learning rate $\alpha = 0.00001$. In addition, the batch size was set to 2, and the balancing hyperparameters in (8) were fixed to 1.0. At the same time, for the loss computation of the segmentation branch during training, the k value of kNN was set to 1 so that the predicted label would be bound to its nearest neighbor of ground truth. The input/output sequence length was set to 5 except for one experiment (Fig. 6), where the length was set to 10.

TABLE 1. Implementation specifications. Each layer contained four types of hyperparameters: search radius (r), the number of sampling neighbors (k), sampling rate, and the number of output channels. SA represents the set abstraction layer. RNN FE represents the flow embedding layer of the RNN unit from PointNet++. FP represents the proposed feature propagation layer, and MLP indicates the element-wise MLP layer.

Layer type	r	k	sample rate	channel
SA	0.5	-	0.5X	128
RNN FE	-	24	-	128
SA	1.0	-	0.5X	128
RNN FE	-	24	-	128
SA	2.0	-	0.5X	128
RNN FE	-	24	-	128
FP	-	3	2X	128
FP	-	3	2X	128
FP	-	3	2X	128
MLP, MLP	-	-	-	3, C

For comparison, we used two baseline models: Input Alignment and PointRNN+RandLA-Net. 1) **Input Alignment.** Because of the order of the input and output point clouds of the network is consistent, this baseline model directly outputs the input coordinates and segmentation labels. 2) **PointRNN+RandLA-Net.** RandLA-Net [4] achieves a state-of-the-art performance in segmentation tasks using real LIDAR points. We applied it to the output point cloud predicted by PointRNN to obtain the corresponding segmentation labels.

B. EVALUATION METRICS

Because the CD (4) and EMD (5) loss functions intuitively describe the distance between the predicted point cloud and the ground truth point cloud, we directly employed these two distance metrics to evaluate the performance of the network prediction branch. For the performance evaluation of the segmentation branch, we defined two metrics: the overall point accuracy and the category point accuracy.

Overall segmentation accuracy (OSA) was used to measure the difference between the overall predicted labels and the aligned ground truth labels generated by kNN (6). Suppose that the predicted labels are L^* and the aligned ground truth labels are $L^\dagger$, then, the OSA can be computed by the following equation:

$$\mathcal{OSA}(L^*, L^\dagger) = \frac{\sum_{i=1}^{|L^\dagger|} I(l^{i*} = l^{i\dagger})}{|L^\dagger|}. \tag{9}$$

Category segmentation accuracy (CSA) measures the difference between the labels and aligned ground truth labels separately in each instance included in the ground truth labels. This ensures that instances consisting of a larger number of points that occupy the most of the scene frame, such as road and sidewalk, with the same weight as the instances that contain a small number of points, such as vehicles and pedestrians when computing accuracy. The CSA is defined

by the following equation.

$$\mathcal{CSA}(L^*, L^\dagger) = \frac{\sum_{j=1}^{|SET(L^\dagger)|} \sum_{i=1}^{|L^\dagger = SET(L^\dagger)^j|} I(l^{i*} = l^{i\dagger})}{|L^\dagger|}, \tag{10}$$

where $SET(\cdot)$ is a function that extracts non-repeating elements from the input set.

C. EVALUATION ON SEMANTICKITTI

1) SEMANTICKITTI DATASET [12]

The SemanticKITTI dataset is currently the most commonly used dynamic point cloud segmentation dataset based on a real-world LIDAR scan. SemanticKITTI describes the street scenes of moving vehicles with 22 annotated LIDAR scan sequences of 43,552 frames of LIDAR data. Among them, we used ten sequences of 23,201 full 3D frames for training, and one sequence of 4,071 frames for verification. Each frame has collected a large-scale point set with up to 100,000 points in the range of 160 × 160m. The computation of the loss function of the network prediction requires a large matrix of size N^2, where N is the length of the predicted point set, in order to determine the neighbor points. Therefore, if we input a larger number of points, the required computational resources will increase exponentially during training, but a broader prediction range will be achieved with a consistent density as well. In order to maintain the balance between the computation and the prediction range, we cropped each frame to the range of 10 × 10m, and randomly sampled 4096 points to the proposed network.

TABLE 2. Segmentation performance on SemanticKITTI. The *P(Phase)* indicates the *i*-th frame output from proposed network.

Phase	Ours		PointRNN + RandLA-Net		Input Alignment	
	OSA	CSA	OSA	CSA	OSA	CSA
1	**89.6%**	78.7%	59.6%	42.4%	87.2%	**79.8%**
2	**87.0%**	**71.8%**	59.6%	40.1%	80.2%	69.7%
3	**84.7%**	**66.0%**	53.8%	38.5%	74.5%	61.7%
4	**82.2%**	**60.8%**	53.8%	36.8%	69.6%	55.0%
5	**79.6%**	**56.7%**	50.0%	35.0%	65.8%	50.2%

2) SEGMENTATION RESULTS

Table 2 summarizes the segmentation performance comparison using the SemanticKITTI dataset. The table shows that the proposed model achieves a higher segmentation accuracy than the baseline models over every phase of the future frames. The table also shows that owing to the characteristics of the prediction of the proposed network, relatively good results can be obtained by directly using the input because the input and output order of the network is consistent. However, the PointRNN+RandLA-Net method that directly input the predicted coordinates into the segmentation network exhibits a poor performance. This is because the prediction network produces errors, which reduce the performance of the segmentation network. The proposed network directly uses the features of the bottom layer of the point cloud, thereby retaining the advantages of consistent order and reducing the negative impact of the prediction errors. We also evaluate the class-wise segmentation accuracy as shown in Table 4. The table shows that our model achieves better accuracy than other methods for most of the classes. Especially, the proposed method outperforms other models for 'car', 'road', and 'sidewalk' classes, which are critical classes for practical applications such as autonomous driving.

TABLE 3. Prediction performance on SemanticKITTI. The *Phase* indicates the *i*-th frame output from the proposed network.

Phase	Ours		PointRNN + RandLA-Net		Input Alignment	
	CD	EMD	CD	EMD	CD	EMD
1	**0.146**	**0.796**	0.172	0.847	0.482	0.782
2	**0.191**	**0.925**	0.205	0.955	0.403	1.048
3	0.246	1.034	**0.237**	**1.032**	0.343	1.245
4	0.309	1.150	**0.292**	**1.129**	0.331	1.401
5	0.379	1.257	**0.350**	**1.221**	0.379	1.522

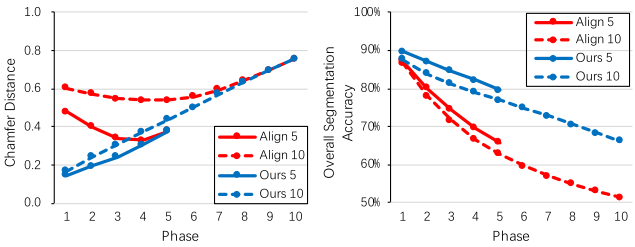

FIGURE 6. The segmentation and prediction performance comparison in terms of various input and output sequence lengths on the SemanticKITTI dataset.

3) PREDICTION RESULTS

Table 3 shows the prediction performance of the proposed network on the SemanticKITTI dataset. Theoretically because the segmentation structure of the proposed network is similar to the PointRNN structure, the RNN unit with the geometric feature constraint will reduce the network prediction performance. However, the performance reduction is not significant, and the prediction in the first two frames even outperforms the baseline network.

4) GENERALIZATION OF SEQUENCE LENGTHS

Fig. 6 shows the performance comparison of the input/output sequence length of 5 and 10 when the k value of kNN is fixed to 1. The result represents that the test with a 10-frame sequence shows a slight performance loss than the test with a 5-frame sequence. This is due to the fact that the model has to learn additional 5 frames with the limited model capacity. However, our model still achieves better performance than the direct input alignment even with the 10-frame sequence.

D. EVALUATION ON ARGOVERSE

1) ARGOVERSE DATASET [13]

Argoverse is a large-scale LIDAR dataset designed for autonomous vehicle perception tasks. Argoverse provides

TABLE 4. Class-wise segmentation performance on SemanticKITTI. Note that, the ratio is the overall average ratio of specific class points over the entire testing set. 'P+R' indicates the PointRNN+RandLA-Net.

Class		Unlabeled	Car	Road	Sidewalk	Building	Fence	Vegetation	Terrain	Pole
Ratio		1.9%	2.8%	21.8%	18.1%	7.6%	8.6%	27.9%	9.4%	1.8%
1	Ours	54.9%	**85.9%**	**95.4%**	**87.6%**	90.0%	**75.9%**	**91.5%**	**68.7%**	58.5%
	P+R	12.0%	44.6%	83.6%	49.9%	56.1%	24.2%	65.5%	29.7%	11.4%
	Align	**69.4%**	84.4%	94.5%	78.8%	**92.7%**	75.4%	90.3%	66.0%	**66.4%**
2	Ours	40.5%	**79.9%**	**94.2%**	**86.0%**	86.4%	**68.4%**	**88.9%**	**62.0%**	40.0%
	P+R	11.8%	35.7%	81.1%	44.6%	56.7%	22.3%	68.1%	29.5%	10.9%
	Align	**53.9%**	71.4%	91.7%	69.3%	**88.3%**	63.8%	84.1%	53.9%	**50.5%**
3	Ours	28.6%	**72.4%**	**93.5%**	**84.0%**	82.7%	**62.8%**	**86.0%**	**58.4%**	25.1%
	P+R	11.6%	30.7%	79.7%	42.0%	54.4%	21.6%	66.8%	29.2%	10.1%
	Align	**42.8%**	59.8%	89.5%	64.2%	**83.7%**	55.2%	78.7%	45.7%	**38.7%**
4	Ours	21.6%	**63.4%**	**92.7%**	**82.1%**	78.1%	**57.3%**	**83.2%**	**54.0%**	14.6%
	P+R	10.7%	26.5%	78.9%	40.1%	51.2%	20.2%	65.7%	29.1%	9.0%
	Align	**33.4%**	50.4%	87.5%	54.8%	**80.0%**	48.6%	74.3%	40.0%	**25.9%**
5	Ours	15.3%	**58.7%**	**91.8%**	**79.3%**	74.6%	**53.6%**	**79.6%**	**49.0%**	8.1%
	P+R	9.8%	22.7%	77.8%	37.6%	47.7%	18.9%	65.1%	28.8%	6.9%
	Align	**26.5%**	43.5%	85.6%	50.3%	**76.5%**	44.5%	70.9%	37.0%	**16.8%**

13,122 LIDAR frames for training and 5,015 frames for verification. Each frame is in the range of 200m and contains up to 107,000 points. However, Argoverse contains only the annotated bounding boxes of tracking objects. To apply this dataset to our task, we use annotated bounding boxes to select points from point cloud and assign their point-wise labels. We assign the 'background' label to remain points except tracking objects. Similarly, to ensure consistency with semanticKITTI, we cropped points in the range of 10×10m from the large-scale point cloud and used random sampling to form the final point set containing 4,096 points.

TABLE 5. Segmentation and prediction performance on Argoverse.

Phase		1	2	3	4	5
Ours	OSA	**98.0%**	**97.5%**	**97.0%**	**96.3%**	**95.7%**
	OSA'	90.6%	88.0%	**85.7%**	**82.3%**	**79.8%**
	CD	**0.301**	**0.305**	**0.368**	**0.448**	**0.549**
	EMD	0.901	1.072	**1.208**	**1.353**	**1.513**
Alignment	OSA	97.5%	96.3%	95.3%	94.2%	94.2%
	OSA'	**93.5%**	**88.3%**	83.2%	78.5%	74.7%
	CD	0.525	0.459	0.427	0.452	0.549
	EMD	**0.771**	**1.102**	1.381	1.601	1.778

2) RESULTS

Table 5 shows the segmentation and prediction performance of the proposed network on the Argoverse dataset. Since most of the points in Argoverse are classified as 'background', the input alignment method can easily achieve a high performance. Nevertheless, the table shows that our model achieves a better performance than the input alignment. In order to clearly evaluate the segmentation performance, we recomputed the OSA accuracy with the 'background' points removed (marked as OSA' in Table 5).

E. ABLATION STUDY

1) GEOMETRIC FEATURE CONSTRAINT

To verify whether the geometric feature constraint can enable the FE layer to obtain local geometric information that contributes to the final segmentation, we introduced a controlled trail in which the FE layers only follow the flow feature constraint. We designed a completely consistent network based on the architecture introduced in Section IV-A. However, the calculation of the loss and gradient propagation was inconsistent with the proposed network. The loss of the segmentation branch and the prediction branch of the controlled trail are calculated independently of each other. In addition, the gradient of the prediction branch propagated to all variables except the MLP layers of the segmentation branch, and the gradient generated by the segmentation branch will only updated its own MLP variables.

TABLE 6. Prediction performance on SemanticKITTI. Our model with a flow feature constraint and local geometric constraint performs better than the controlled trail with only a flow feature constraint, which only works for points prediction.

Phase		1	2	3	4	5
Ours	OSA	**89.6%**	**87.0%**	**84.7%**	**82.2%**	**79.6%**
	CD	0.146	0.191	0.246	0.309	0.379
	EMD	0.796	0.925	1.034	1.150	1.257
Control	OSA	59.6%	59.6%	53.7%	53.7%	50.0%
	CD	0.135	0.173	0.225	0.281	0.342
	EMD	0.777	0.906	1.019	1.117	1.219

According to the experimental results shown in Table 6, the geometric feature constraint can activate the extraction ability of the local geometric feature of the FE layer, improving the segmentation accuracy. Although it can sacrifices a part of the flow feature extraction performance while obtaining local geometric features, using this constraint increases segmentation accuracy by at least 30% while reducing the prediction accuracy by only approximately 0.5%. Therefore, it can be inferred that the strategy of introducing a geometric feature constraint is effective.

2) NUMBER OF NEIGHBOR POINTS

As we group more neighbor points, the SA layer can extract stronger local features. Therefore, the number of neighbor

points will significantly affect the performance of prediction and segmentation. However, increasing neighbours requires more computational cost. To evaluate the effect of the neighbor size, we conducted experiments with the number of neighbors as 8, 16, and 24, as shown in Fig. 7. Compared to the model with 8 neighbors, the model with 16 neighbours shows a significant improvement in terms of both segmentation and prediction performance. However, as we increase the size further, we get only marginal performance enhancement. Therefore, we set the number of neighbors as 24 for local feature extraction in this work.

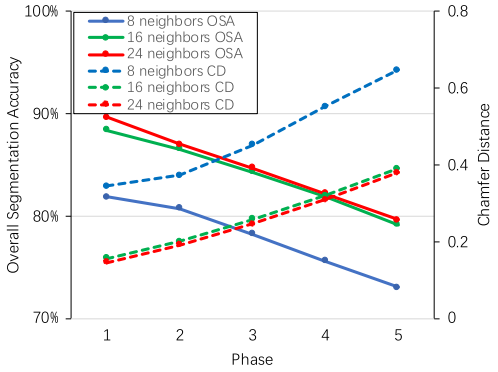

FIGURE 7. The segmentation and prediction performance comparison in terms of various number of neighbours on SemanticKITTI dataset.

TABLE 7. Runtime and model size. The input sequence length is 5 and the number of input points is 4096. The speed is measured by the average frame rate over the entire SegmentKITTI dataset.

Methods	Speed (fps)	Parameters
PointNet [3]	21.2	0.8M
PointNet++ [7]	0.41	0.97M
RandLA-Net [4]	22.0	1.24M
Ours	5.17	0.39M

3) RUNTIME AND MODEL SIZE

As the joint prediction and segmentation task is newly introduced task, there is not a similar reference to compare. Therefore, we compared computational time with representative static segmentation models, as shown in Table 7.The computational time of the proposed model for segmentation and classification is 0.967s for one stream (5 frames), which is 5.17 frames per second (fps). Because our model performs prediction as well as segmentation, it takes more computation time than other static segmentation models. Nevertheless, real time applications can benefit from our approach since it can predict the feature scene and respond in advance. It should also be noted that the proposed model requires less parameters than other models even with the added prediction capability.

VI. CONCLUSION

In this paper, we introduce a joint deep neural network architecture for simultaneously segmenting and predicting point cloud of the future scenes, as the first work that shows the success in solving a joint segmentation and prediction task with the future point cloud. To implement the joint segmentation and prediction network, we employed the FE layer for both local geometric feature and flow features extraction and proposed a new training method for future point segmentation. Based on the experiments with various real-world LIDAR scan benchmarks, the proposed network shows better performance than other baseline methods. In the future, we will focus on accelerating the proposed network while improving the prediction performance that benefit the segmentation as well.

REFERENCES

[1] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58443–58469, 2020.
[2] B. Schwarz, "Mapping the world in 3D," *Nature Photon.*, vol. 4, no. 7, pp. 429–430, Jul. 2010.
[3] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 652–660.
[4] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham, "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, Jun. 2020, pp. 11108–11117.
[5] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen, "PointCNN: Convolution on *X*-transformed points," in *Proc. Adv. Neural Inf. Process. Syst.*, 2018, pp. 820–830.
[6] J. Li, B. M. Chen, and G. H. Lee, "SO-Net: Self-organizing network for point cloud analysis," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 9397–9406.
[7] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1–14.
[8] H. Fan and Y. Yang, "PointRNN: Point recurrent neural network for moving point cloud processing," 2019, *arXiv:1910.08287*. [Online]. Available: http://arxiv.org/abs/1910.08287
[9] X. Weng, J. Wang, S. Levine, K. Kitani, and N. Rhinehart, "Unsupervised sequence forecasting of 100,000 points for unsupervised trajectory forecasting," Jan. 2020, *arXiv:2003.08376*. [Online]. Available: http://arxiv.org/abs/2003.08376
[10] X. Liu, C. R. Qi, and L. J. Guibas, "FlowNet3D: Learning scene flow in 3D point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 529–537.
[11] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 1–9.
[12] J. Behley, M. Garbade, A. Milioto, J. Quenzel, S. Behnke, C. Stachniss, and J. Gall, "SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2019, pp. 9297–9307.
[13] M.-F. Chang, D. Ramanan, J. Hays, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, and S. Lucey, "Argoverse: 3D tracking and forecasting with rich maps," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2019, pp. 8748–8757.
[14] Y. Xie, J. Tian, and X. X. Zhu, "Linking points with labels in 3D: A review of point cloud semantic segmentation," *IEEE Geosci. Remote Sens. Mag.*, vol. 8, no. 4, pp. 38–59, Dec. 2020.
[15] J. Chen, Y. K. Cho, and Z. Kira, "Multi-view incremental segmentation of 3-D point clouds for mobile robots," *IEEE Robot. Autom. Lett.*, vol. 4, no. 2, pp. 1240–1246, Apr. 2019.
[16] H.-Y. Meng, L. Gao, Y.-K. Lai, and D. Manocha, "VV-Net: Voxel VAE net with group convolutions for point cloud segmentation," in *Proc. IEEE Int. Conf. Comput. Vis.*, Oct. 2019, pp. 8500–8508.
[17] D. Maturana and S. Scherer, "VoxNet: A 3D convolutional neural network for realtime object recognition," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*. IEEE, 2015.
[18] Y. Wang, T. Shi, P. Yun, L. Tai, and M. Liu, "PointSeg: Real-time semantic segmentation based on 3D LiDAR point cloud," 2018, *arXiv:1807.06288*. [Online]. Available: http://arxiv.org/abs/1807.06288

[19] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet ++: Fast and accurate LiDAR semantic segmentation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Nov. 2019, pp. 4213–4220.
[20] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR point cloud," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 1887–1893.
[21] X. Liu, Z. Han, Y.-S. Liu, and M. Zwicker, "Point2Sequence: Learning the shape representation of 3D point clouds with an attention-based sequence to sequence network," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33., 2019, pp. 8778–8785.
[22] Q. Huang, W. Wang, and U. Neumann, "Recurrent slice networks for 3D segmentation of point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2626–2635.
[23] X. Ye, J. Li, H. Huang, L. Du, and X. Zhang, "3D recurrent neural networks with context fusion for point cloud semantic segmentation," in *Proc. Eur. Conf. Comput. Vis. (ECCV)*, 2018, pp. 403–417.
[24] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph CNN for learning on point clouds," *ACM Trans. Graph.*, vol. 38, no. 5, pp. 1–12, 2019.
[25] K. Zhang, M. Hao, J. Wang, C. W. de Silva, and C. Fu, "Linked dynamic graph CNN: Learning on point cloud via linking hierarchical features," 2019, *arXiv:1904.10014*. [Online]. Available: http://arxiv.org/abs/1904.10014
[26] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3D point clouds," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 40–49.
[27] L. Yu, X. Li, C.-W. Fu, D. Cohen-Or, and P.-A. Heng, "PU-Net: Point cloud upsampling network," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 2790–2799.
[28] W. Yuan, T. Khot, D. Held, C. Mertz, and M. Hebert, "PCN: Point completion network," in *Proc. Int. Conf. 3D Vis. (3DV)*, Sep. 2018, pp. 728–737.
[29] X. Weng, J. Wang, S. Levine, K. Kitani, and N. Rhinehart, "Sequential forecasting of 100,000 points," 2020, *arXiv:2003.08376*. [Online]. Available: https://arxiv.org/abs/2003.08376
[30] Y. Yang, C. Feng, Y. Shen, and D. Tian, "FoldingNet: Point cloud auto-encoder via deep grid deformation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jun. 2018, pp. 206–215.
[31] W. Cheng and S. Lee, "Point auto-encoder and its application to 2D-3D transformation," in *Proc. Int. Symp. Vis. Comput.* Cham, Switzerland: Springer, 2019, pp. 66–78.
[32] H. Fan, H. Su, and L. Guibas, "A point set generation network for 3D object reconstruction from a single image," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, Jul. 2017, pp. 605–613.
[33] Y. Rubner, C. Tomasi, and L. J. Guibas, "The Earth mover's distance as a metric for image retrieval," *Int. J. Comput. Vis.*, vol. 40, no. 2, pp. 99–121, Nov. 2000.
[34] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*. [Online]. Available: http://arxiv.org/abs/1412.6980

CHENG WENCAN (Graduate Student Member, IEEE) received the B.S. degree in measurement and control technology from the Nanjing University of Information Science and Technology, Nanjing, China, in 2017, and the M.S. degree in artificial intelligence from Sungkyunkwan University, Suwon, South Korea, in 2020. His research interests include efficient hardware and 3D structural data processing.

JONG HWAN KO (Member, IEEE) received the dual B.S. degrees in computer science and engineering and mechanical and aerospace engineering and the M.S. degree in electrical engineering and computer science from Seoul National University, and the Ph.D. degree from the School of Electrical and Computer Engineering, Georgia Tech, in 2018. He joined Sungkyunkwan University (SKKU), South Korea, as an Assistant Professor. During his seven years research experience at the Agency for Defense Development (ADD), South Korea, he conducted advanced research on the design and performance analysis of military wireless sensor networks. His research interests include design of low-power image sensor systems and deep learning accelerators for efficient image/audio processing. He has received the Best Paper Award from the International Symposium on Low Power Electronics and Design (ISLPED), in 2016.