

# MoNet: Motion-based Point Cloud Prediction Network

Fan Lu<sup>1</sup>, Guang Chen<sup>1,2,✉</sup>, Yinlong Liu<sup>2</sup>, Zhijun Li<sup>3</sup>, Sanqing Qu<sup>1</sup>, Tianpei Zou<sup>1</sup>  
<sup>1</sup>Tongji University, <sup>2</sup>Technische Universität München,  
<sup>3</sup>University of Science and Technology of China

lufan@tongji.edu.cn, guangchen@tongji.edu.cn, Yinlong.Liu@tum.de  
zjli@ieee.org, 2011444@tongji.edu.cn, 2011459@tongji.edu.cn

## Abstract

*Predicting the future can significantly improve the safety of intelligent vehicles, which is a key component in autonomous driving. 3D point clouds accurately model 3D information of surrounding environment and are crucial for intelligent vehicles to perceive the scene. Therefore, prediction of 3D point clouds has great significance for intelligent vehicles, which can be utilized for numerous further applications. However, due to point clouds are unordered and unstructured, point cloud prediction is challenging and has not been deeply explored in current literature. In this paper, we propose a novel motion-based neural network named MoNet. The key idea of the proposed MoNet is to integrate motion features between two consecutive point clouds into the prediction pipeline. The introduction of motion features enables the model to more accurately capture the variations of motion information across frames and thus make better predictions for future motion. In addition, content features are introduced to model the spatial content of individual point clouds. A recurrent neural network named Motion-RNN is proposed to capture the temporal correlations of both features. Besides, we propose an attention-based motion align module to address the problem of missing motion features in the inference pipeline. Extensive experiments on two large scale outdoor LiDAR datasets demonstrate the performance of the proposed MoNet. Moreover, we perform experiments on applications using the predicted point clouds and the results indicate the great application potential of the proposed method.*

## 1. Introduction

Predicting the future is an essential capability of intelligent vehicles, which allows for anticipation of what will happen in the future and can significantly improve the safety of autonomous driving [26, 21, 40, 14, 2]. 3D point clouds can provide accurate 3D modeling of the surrounding environment and have been widely used in numerous perception

applications (e.g., object detection [28, 29, 37, 38], semantic segmentation [10, 19, 35, 39]). Compared to existing video prediction [11, 14, 32, 33], the prediction of 3D point clouds can provide an opportunity for a better understanding of future scenes [8], which can help the intelligent vehicles perceive the environment and make decisions in advance. However, 3D point clouds are unordered and unstructured [23], which makes the prediction of 3D point clouds more challenging than video prediction and there is little exploration of point cloud prediction in the current literature.

Based on the above observations, we study a problem named *Point Cloud Prediction* in this paper. Concretely, given a sequence contains  $T$  frames of point clouds  $\{P_1, \dots, P_T\}$ , point cloud prediction aims to predict  $T_p$  future frames  $\{P_{T+1}, \dots, P_{T+T_p}\}$ . The prediction of future point clouds can be formulated as a per-point motion prediction problem. Intuitively, the prediction model should be capable of capturing the variations of motion information across point cloud frames, which can benefit from explicitly estimating motion features between two point clouds. However, the introduction of motion features in point cloud prediction has not been explored in existing methods [6, 34].

To address the above problem, we propose a motion-based neural network named MoNet for point cloud prediction. Different from previous works [6, 34], the key idea of MoNet is to integrate motion information between two consecutive point clouds into point cloud prediction. Motion features between two point clouds are extracted using a motion encoder, which allows the network to accurately capture the variations of motion information. Content features of an individual frame are introduced to model the spatial content of a point cloud itself, which helps preserve spatial structures of point clouds. The combination of content and motion features can exploit complementary advantages and results in better performance. To capture the temporal correlations of both features across frames, we propose a recurrent neural network (RNN) named MotionRNN. Unlike standard RNN and its variants like Long Short-Term Memory (LSTM) [9] and Gated Recurrent Unit (GRU) [3] which

can only process one-dimensional features, the proposed MotionRNN associates the states with the coordinates of points to maintain the spatial structure of point clouds. During the inference pipeline, an attention-based motion align module is proposed to estimate the motion features to address the absence of the next point cloud. The overall architecture of the proposed MoNet is shown in Fig. 1.

To verify the feasibility of the proposed method, we perform extensive experiments on two large scale outdoor LiDAR point cloud datasets, namely KITTI odometry dataset [7] and Argoverse dataset [1]. Both qualitative and quantitative results show that the proposed MoNet significantly outperforms baseline methods and is capable of effectively predicting the future point clouds. Besides, the experiments on applications indicate the great application potential of the proposed method.

To summarize, our main contributions are as follows:

- A novel motion-based neural network named MoNet is proposed for point cloud prediction, which can effectively predict the future point clouds.
- Motion features are explicitly extracted and integrated into point cloud prediction and the combination of motion and content features results in better performance. Besides, MotionRNN is proposed to model the temporal correlations of both features.
- An attention-based motion align module is proposed in the inference pipeline to estimate the motion features without future point clouds.

## 2. Related works

We briefly review the most related works, including video prediction, sequential point clouds processing and point cloud prediction.

**Video prediction** Video prediction aims to predict future images given a sequence of previous frames. ConvLSTM [36] introduced 2D convolutions into LSTM to extract visual representations, which has been a seminal work in video prediction. Based on [36], Wang et al. [33] proposed ST-LSTM to memorize spatial appearance and temporal variations simultaneously. Video pixel network (VPN) [12] utilized PixelCNNs to directly estimate the discrete joint distribution of the raw pixel values for video prediction. Villegas et al. [31] proposed to decompose motion and content and independently capture each information stream. Eidetic 3D LSTM (E3D-LSTM) [32] was proposed to integrate 3D convolutions into RNNs for video prediction and exploit complementary advantages. Generative adversarial network (GAN) has also been widely applied in video prediction to improve the quality of generated images [15, 14].

**Sequential point clouds processing** Processing sequential point clouds is challenging due to that point clouds are disordered and unstructured. FlowNet3D [17] utilized flow

embedding layer to model the correspondence of points between two point clouds, which is a representative work to process consecutive point clouds using deep learning-based method. Liu et al. [18] proposed MetroNet, which learns to aggregate information from spatiotemporal neighbor points to learn representations for dynamic point cloud sequences. In [27], a cross-frame global attention module and local interpolation module were proposed to capture spatial and temporal information in point cloud sequences for point cloud semantic segmentation. In [20], a weight-shared LSTM model named PointLSTM was proposed to update state information for neighbor point pairs to perform gesture recognition. Choy et al. [4] proposed 4D spatio-temporal convolutional network for 3D-video perception.

**Point cloud prediction** Point cloud prediction has not been deeply explored in literature. Fan et al. [6] proposed PointRNN to model temporal information across point clouds for future point clouds prediction. They proposed a point-based spatiotemporal-local correlation named point-rnn to replace concatenation operation in standard RNN to process moving point clouds. Very recently, Weng et al. [34] adopted an encoder-decoder architecture to predict future point clouds and utilizes the predicted point clouds to perform trajectory prediction.

## 3. Methodology

### 3.1. Overall architecture

Given a point cloud sequence  $\{P_1, \dots, P_T\} \in \mathbb{R}^{N \times 3}$ , the goal of point cloud prediction is to predict  $T_p$  future point clouds  $\{P_{T+1}, \dots, P_{T+T_p}\} \in \mathbb{R}^{N \times 3}$ . To achieve that, we propose a novel motion-based neural network named MoNet and the overall architecture is shown in Fig. 1. MoNet can be divided into two pipelines, namely embedding pipeline and inference pipeline. The input point clouds are firstly inputted into the embedding pipeline to extract the content and motion features and capture the temporal correlations across point clouds. Then the embedded features are passed into the inference pipeline to predict the future point cloud frames. We denote the point coordinates, content features and motion features of frame  $t$  in layer  $l$  as  $X_t^l \in \mathbb{R}^{N_l \times 3}$ ,  $E_t^l \in \mathbb{R}^{N_l \times d_t^E}$  and  $M_t^l \in \mathbb{R}^{N_l \times d_t^M}$ , where  $N_l$  is the number of points in layer  $l$ ,  $d_t^E$  and  $d_t^M$  are number of channels of content and motion features, respectively. We will describe the two pipelines in detail below.

### 3.2. Embedding

As shown in the left part of Fig. 1, the key component of the embedding pipeline is the MECell (*i.e.*, Motion Embedding Cell). As we mentioned before, content features of an individual frame and motion features of consecutive point clouds can both contribute to point cloud prediction. However, previous works only model the temporal infor-

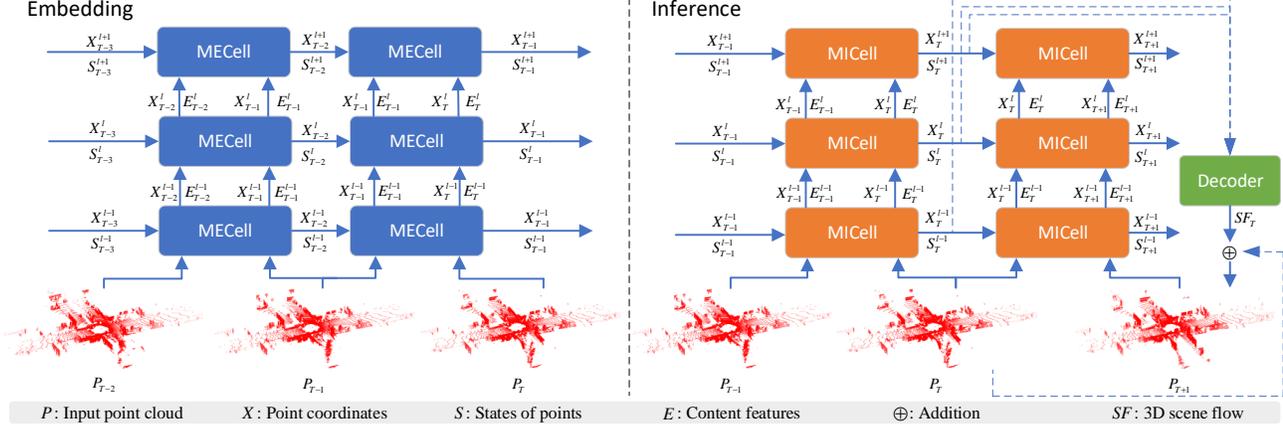


Figure 1. Overall architecture of the proposed MoNet. The left part displays the embedding pipeline, where the MECell (Motion Embedding Cell) firstly extracts content features and motion features and then captures temporal information across point clouds. The right part shows the inference pipeline and the MICell (Motion Inference Cell) is adopted to replace MECell to predict future frames.

mation of point cloud sequence and do not explicitly extract the motion features between point clouds. As shown in the left part of Fig. 2, given content features and point coordinates  $(X_t^{l-1}, E_t^{l-1})$ ,  $(X_{t+1}^{l-1}, E_{t+1}^{l-1})$  of two consecutive point clouds from the last layer  $l-1$ , the content encoder is firstly adopted to extract point coordinates and content features  $(X_t^l, E_t^l)$ ,  $(X_{t+1}^l, E_{t+1}^l)$  of current layer  $l$ . Then motion encoder is followed to generate motion features  $M_t^l$  corresponding to frame  $t$ , which models the per-point motion information from frame  $t$  to frame  $t+1$ . After that,  $M_t^l$ ,  $E_t^l$ ,  $X_t^l$ , the states  $S_{t-1}^l$  and coordinates  $X_{t-1}^l$  from last frame  $t-1$  are passed into a recurrent neural network named MotionRNN to simultaneously capture the temporal information of motion features and content features across point clouds and generate the states  $S_t^l$  of current frame  $t$ .

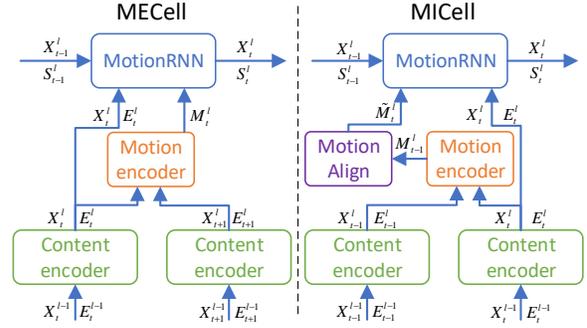


Figure 2. Left: MECell (Motion Embedding Cell). Right: MICell (Motion Inference Cell).

### 3.2.1 Content encoder

We adopt a PointNet++[24]-like structure to encode the content features of point clouds. Given  $X_t^{l-1}$  and  $E_t^{l-1}$  from the last layer, firstly  $N_t$  points  $X_t^l$  are sampled from  $X_t^{l-1}$  using Furthest Point Sampling (FPS). For each point  $x_i \in X_t^l$ ,  $k$  neighbor points  $\{x_i^1, \dots, x_i^k\}$  are searched in  $X_t^{l-1}$  around  $x_i$  using  $k$ -nearest-neighbors ( $k$ NN) method to generate a cluster. The relative coordinates  $\{x_i^1 - x_i, \dots, x_i^k - x_i\}$  and the relative distances  $\{\|x_i^1 - x_i\|, \dots, \|x_i^k - x_i\|\}$  are calculated as geometric features of the cluster, where  $\|\cdot\|$  denotes Euclidean distance. The geometric features are then concatenated with content features  $\{e_i^1, \dots, e_i^k\}$  to generate a feature map, where  $e_i^j$  is the content feature corresponding to  $x_i^j$ . After that, the feature maps of all clusters are passed into Shared Multilayer Perceptron (Shared-MLP) with maxpool layer to generate content features  $E_t^l$  of the current layer  $l$ .

### 3.2.2 Motion encoder

Motion encoder is proposed to model the motion information between two consecutive point clouds. The input of motion encoder are the point coordinates and content features  $(X_t^l, E_t^l)$ ,  $(X_{t+1}^l, E_{t+1}^l)$  of two point clouds, and the output is the motion features from  $X_t^l$  to  $X_{t+1}^l$ . For each point  $x_i$  in  $X_t^l$ , we search  $k$  neighbor points in  $X_{t+1}^l$  and then use a similar strategy as content encoder to generate geometric features. The feature map is a concatenation of geometric features, content features of neighbor points in  $X_{t+1}^l$  and content features of the center point  $x_i$ . After that, Shared-MLP with maxpool layer are followed to generate motion features  $M_t^l$ . Intuitively, the content features of two point clouds and geometric features between two frames all contribute to the generated motion features, which enables the motion encoder to capture the changes of contents and point coordinates across two point clouds. Thus, motion features of frame  $t$  can be considered as a representation of motion information from frame  $t$  to  $t+1$ .

### 3.2.3 MotionRNN

General RNN models like LSTM and GRU can only process one-dimensional features. However, spatial information is important in the representation of point clouds. A one-dimensional global feature is hard to preserve the spatial structure and local details of point clouds thus is not applicable for large scale point cloud prediction. Based on the above consideration, we propose a novel recurrent neural network named MotionRNN for point cloud prediction. The key idea of MotionRNN is that the states correspond to point coordinates one by one and the content and motion features are combined to update the states. The inputs of MotionRNN are the points  $X_{t-1}^l$  and states  $S_{t-1}^l$  from last frame, the points  $X_t^l$ , content features  $E_t^l$  and motion features  $M_t^l$  of current frame and the output are the points  $X_t^l$  and states  $S_t^l$  of current frame.

We provide two versions of MotionRNN, namely MotionGRU and MotionLSTM. Taken MotionLSTM as an example, the states  $S_t^l$  consists of hidden states  $H_t^l \in \mathbb{R}^{N_t \times d_t^S}$  and cell states  $C_t^l \in \mathbb{R}^{N_t \times d_t^S}$ . For each point in  $X_t^l$ ,  $k$  neighbor points are searched in  $X_{t-1}^l$  and the similar strategy in content encoder is adopted to generate the geometric features  $\bar{G}_t^l$  and also clusters. Denote the hidden state and cell state of clusters as  $\bar{H}_{t-1}^l$  and  $\bar{C}_{t-1}^l$ , then the updates for MotionLSTM at  $t$ -th step in layer  $l$  can be formulated as:

$$\begin{aligned}
 I_t^l &= \sigma(\text{maxpool}(\text{MLP}([\bar{G}_t^l, \bar{H}_{t-1}^l, M_t^l, E_t^l]))) \\
 F_t^l &= \sigma(\text{maxpool}(\text{MLP}([\bar{G}_t^l, \bar{H}_{t-1}^l, M_t^l, E_t^l]))) \\
 O_t^l &= \sigma(\text{maxpool}(\text{MLP}([\bar{G}_t^l, \bar{H}_{t-1}^l, M_t^l, E_t^l]))) \\
 \hat{C}_{t-1}^l &= \text{maxpool}(\text{MLP}([\bar{G}_t^l, \bar{C}_{t-1}^l])) \\
 \tilde{C}_t^l &= \tanh(\text{maxpool}(\text{MLP}([\bar{G}_t^l, \bar{H}_{t-1}^l, M_t^l, E_t^l]))) \\
 C_t^l &= F_t^l \odot \hat{C}_{t-1}^l + I_t^l \odot \tilde{C}_t^l \\
 H_t^l &= O_t^l \odot \tanh(C_t^l)
 \end{aligned} \tag{1}$$

where  $\sigma(\cdot)$ ,  $[\cdot]$  and  $\odot$  represent Sigmoid function, concatenation operation and Hadamard product, respectively. Similarly, the updates for MotionGRU can be represented as Eq. 2, where the states  $S_t^l$  consists of only hidden states  $H_t^l$ .

$$\begin{aligned}
 Z_t^l &= \sigma(\text{maxpool}(\text{MLP}([\bar{G}_t^l, \bar{H}_{t-1}^l, M_t^l, E_t^l]))) \\
 R_t^l &= \sigma(\text{maxpool}(\text{MLP}([\bar{G}_t^l, \bar{H}_{t-1}^l, M_t^l, E_t^l]))) \\
 \hat{H}_{t-1}^l &= \text{maxpool}(\text{MLP}([\bar{G}_t^l, \bar{H}_{t-1}^l])) \\
 \tilde{H}_t^l &= \tanh(\text{MLP}([R_t^l \odot \hat{H}_{t-1}^l, M_t^l, E_t^l])) \\
 H_t^l &= Z_t^l \odot \hat{H}_{t-1}^l + (1 - Z_t^l) \odot \tilde{H}_t^l
 \end{aligned} \tag{2}$$

Compared to standard RNN models, the proposed MotionRNN corresponds states to coordinates one by one. This representation can well maintain the spatial structure of

point clouds, thus is more applicable for point cloud prediction. Different from PointRNN [6], the motion features and content features are incorporated into the recurrent neural network, which enables the network to simultaneously capture the variations of motion information and temporal correlations of spatial contents across frames.

### 3.3. Inference

In the embedding pipeline, the motion features and content features are extracted and the temporal correlations are modeled using the proposed MotionRNN. In general prediction models, the process in the inference pipeline is exactly the same as that in the embedding pipeline, where the output states of the last frame of the embedding pipeline are duplicated to the first cell of the inference pipeline to predict the first future frame [30, 36]. However, this general strategy is not applicable to our method. Noting that the motion features of frame  $t$  are generated using the content features of frame  $t$  and  $t + 1$ . However, the next frame  $T + 1$  does not exist for the last frame  $T$  of the embedding pipeline so that the motion features of frame  $T$  is not available.

To address the above problem, we propose MICell (*i.e.*, Motion Inference Cell) in the inference pipeline to replace the MECell in the embedding pipeline. The architecture of MICell can be seen in the right part of Fig. 2. Given  $(X_{t-1}^{l-1}, E_{t-1}^{l-1})$  and  $(X_t^{l-1}, E_t^{l-1})$ , the content encoder and motion encoder are the same as that in MECell. However, to overcome the lack of motion features corresponding to frame  $t$ , a motion align module is applied after the motion encoder to align the motion features of frame  $t - 1$  to frame  $t$ , which is the only difference between MECell and MICell. Then the estimated motion features  $\tilde{M}_t^l$  and content features  $E_t^l$  with coordinates  $X_t^l$  are input into MotionRNN to generate the states  $S_t^l$ . Finally, the decoder is followed to decode the hidden states of frame  $t$  to the 3D scene flow  $SF_t$  of points in  $P_t$  and the future point cloud  $P_{t+1}$  can be calculated as  $P_{t+1} = P_t + SF_t$ . Here we adopt the Feature Propagation module in PointNet++ [24] as the decoder and decode the hidden states in a coarse-to-fine manner.

#### 3.3.1 Motion align

Motion align module is introduced to estimate motion features  $\tilde{M}_t^l$  of current frame  $t$  from the motion features of last frame  $t - 1$ . Noting that the motion features are corresponding to the points coordinates one-by-one. To estimate the motion features, we make two basic assumptions based on the fact that the time between two consecutive point clouds is small: (1) From frame  $t - 1$  to frame  $t$ , the movement of a point is limited; (2) From frame  $t - 1$  to frame  $t$ , the motion feature of a point will not change significantly. Based on the above two assumptions, the motion feature of a point in the current frame can be estimated given the motion features of

its neighbor points in the previous frame.

Here we adopt an attention mechanism to perform the estimation of motion features. The network architecture of the proposed motion align module is shown in Fig. 3. Taken a point  $x_t^l$  in  $X_t^l$  as an example,  $k$  neighbor points are searched in  $X_{t-1}^l$  and the geometric features  $\bar{g}_t^l \in \mathbb{R}^{k \times 4}$  are extracted using the same operation as described in the content encoder. The geometric features are then concatenated with motion features  $\bar{m}_{t-1}^l \in \mathbb{R}^{k \times d_t^M}$  of the neighbor points to produce a feature map. Shared-MLP and max-pool layer with a Softmax function are followed to predict attentive weight for each neighbor point. The motion feature  $\tilde{m}_t^l$  for a point in frame  $t$  can be represented as the weighted-sum of the motion features of the neighbor points in frame  $t - 1$ . This operation will be applied on all points in  $X_t^l$  to generate motion features  $\tilde{M}_t^l$ . Intuitively, the attention mechanism can assign higher weights to the neighbor points that are more likely to be corresponding to the center point  $x_t^l$ . As a result, the proposed motion align module can reasonably estimate the motion features of frame  $t$ .

### 3.3.2 Loss

We adopt Chamfer Distance (CD) between the ground truth point cloud  $P_t \in \mathbb{R}^{N \times 3}$  and the predicted point cloud  $\hat{P}_t \in \mathbb{R}^{N \times 3}$  as the loss function. Chamfer Distance [5] is a commonly used metric to measure the similarity between two point clouds, which can be formulated as:

$$\mathcal{L} = \frac{1}{N} \sum_{\hat{x}^i \in \hat{P}_t} \min_{x^j \in P_t} \|\hat{x}^i - x^j\| + \frac{1}{N} \sum_{x^j \in P_t} \min_{\hat{x}^i \in \hat{P}_t} \|x^j - \hat{x}^i\| \quad (3)$$

## 4. Experiments

### 4.1. Datasets

The proposed method is evaluated on two large scale outdoor LiDAR datasets, namely KITTI odometry dataset [7] and Argoverse dataset [1]. KITTI dataset contains 11 sequences (00-11) with ground truth, and we use sequence 00 to 05 to train the network, 06 to 07 to validate and 08 to 10 to test. We use the 3D tracking branch in Argoverse dataset and follow the default splits to train, validate and test the networks. For each dataset, we sample 10 consecutive point clouds as a sequence, where the first 5 frames are the input point clouds and the last 5 frames are used as the ground truth of predicted point clouds.

### 4.2. Baseline methods

We compare our method with several baseline methods to demonstrate the performance. (1) PointRNN (LSTM) and PointRNN (GRU): PointRNN is proposed in [6] and

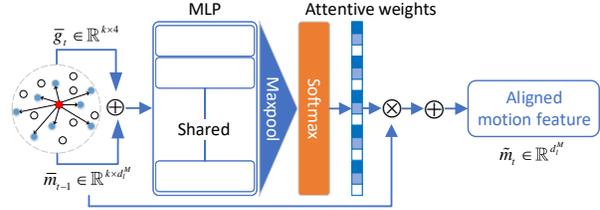


Figure 3. The architecture of the proposed motion align module. The left circle represents a  $k$ -nearest-neighbors cluster, where the red point is a point in frame  $t$ , blue points are neighbor points in frame  $t - 1$  and white points represent other points in frame  $t$ .

we call the two variations using LSTM and GRU as PointRNN (LSTM) and PointRNN (GRU). The networks are re-trained on the two datasets due to the different pre-processing of point clouds in [6]. (2) PointNet++ (LSTM) and PointNet++ (GRU): We define two additional baselines based on the encoder-decoder architecture of PointNet++ [24]. Firstly the input point clouds are encoded into one-dimensional global features. After that, LSTM or GRU is utilized to model the temporal information and the generated global feature is decoded into the predicted per-point motions to generate future point clouds. (3) Scene flow: We utilize the 3D scene flow estimation network FlowNet3D [17] to estimate the 3D scene flow between the last two input point clouds and use the estimated scene flow to predict the future point clouds. FlowNet3D is finetuned on the two datasets to achieve better performance.

### 4.3. Implementation details

Noting that we provide two versions of MotionRNN named MotionLSTM and MotionGRU, and MoNet using the two versions are denoted as MoNet (LSTM) and MoNet (GRU), respectively. We stack multi-layer MECell and MI-Cell to construct a hierarchical structure and we use a 3-layer architecture in our implementation as shown in Fig. 1. The points and states of last frame for the first MECell are initialized to zero and the input content features of the first layer (*i.e.*, the input point clouds) are also initialized to zero. The network is implemented using PyTorch [22] and Adam [13] is used as the optimizer. The point clouds are downsampled to 16384 points using random sampling during training. The network is trained and evaluated on NVIDIA GeForce RTX 2080Ti. More details about the network structure are provided in the supplementary materials.

### 4.4. Qualitative evaluation

We provide several qualitative evaluation results on KITTI odometry dataset in Fig. 4 and the point clouds are downsampled to 32768 points during pre-processing. The images from left to right display the future point clouds from frame  $t = 1$  to  $t = 5$  and different rows represent the results of different methods. For a fair comparison, all of the methods using recurrent neural network are based on GRU.

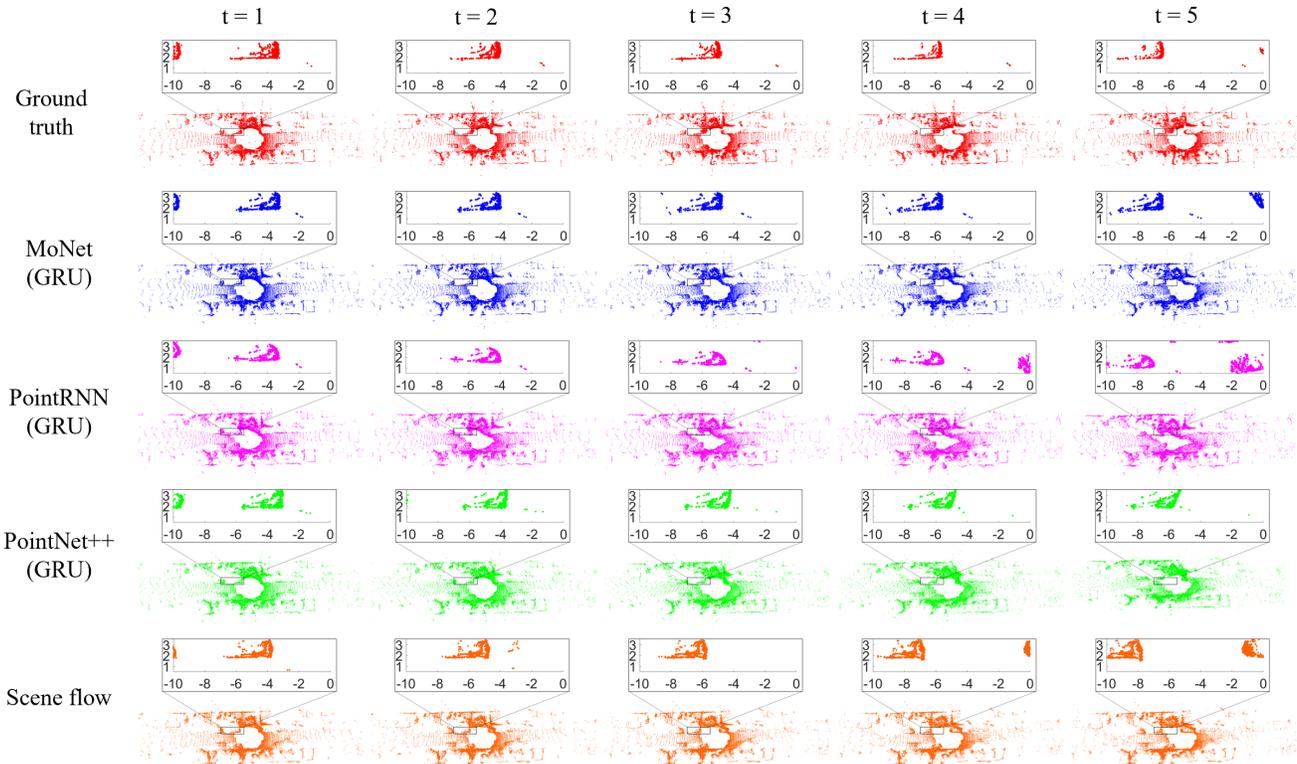


Figure 4. Qualitative visualization of predicted point clouds of different methods on KITTI odometry dataset. From left to right are the predicted future point clouds from  $t = 1$  to  $t = 5$ . Rows from top to bottom display the ground truth point clouds and the predicted results of MoNet (GRU), PointRNN (GRU), PointNet++ (GRU) and Scene flow.

Besides, we zoom in an area containing the point cloud of a vehicle for better visualization. According to Fig. 4, the point clouds predicted by our MoNet (GRU) are more consistent with the ground truth point clouds than other baseline methods. From the zoomed-in point clouds, the proposed method precisely predicts the position of the vehicle and the geometry of the point cloud is well preserved. Compared with our method, the point clouds of the vehicle from PointNet++ (GRU) are deformed, which is due to the poor representative ability of one-dimensional global features. PointRNN (GRU) can not correctly predict the position of the vehicle due to the lack of explicit modeling of motion information. Based on the qualitative results, the proposed method can precisely predict the motion of the point clouds and well preserve the details of point clouds.

## 4.5. Quantitative evaluation

### 4.5.1 Evaluation metrics

We adopt two metrics to evaluate the performance, namely Chamfer Distance (CD)[5] and Earth Mover’s Distance (EMD) [25]. CD is described previously in Eq 3. EMD is also a commonly used metric to compare two point clouds, which is implemented by solving a linear assignment problem between two point clouds. Given two point clouds

$P_t \in \mathbb{R}^{N \times 3}$  and  $\hat{P}_t \in \mathbb{R}^{N \times 3}$ , EMD can be calculated as:

$$EMD = \min_{\phi: \hat{P}_t \rightarrow P_t} \frac{1}{N} \sum_{\hat{x}^j \in \hat{P}_j} \|\hat{x} - \phi(\hat{x})\| \quad (4)$$

where  $\phi: \hat{P}_t \rightarrow P_t$  is a bijection from  $\hat{P}_t$  to  $P_t$ . Due to the high computational complexity of EMD, the point clouds are downsampled to 16384 using random sampling during pre-processing in quantitative evaluation.

### 4.5.2 Chamfer Distance

We calculate the Chamfer Distance (CD) between predicted point clouds and ground truth ones on 5 future frames on KITTI and Argoverse dataset and display the results in Fig. 5. For better comparison, we display the results using LSTM and GRU on the left two images and right two images in Fig. 5, respectively. The CD of our methods is lower than other baseline methods by an obvious margin on two datasets. For example, the average CD on KITTI dataset of MoNet (LSTM) is about 30% smaller than the best baseline PointRNN (LSTM). The Chamfer Distance increases as the number of predicted frames increases, which is due to the increase in uncertainty. However, the growth of CD of our method is much slower than other methods, which demonstrates the robustness of the proposed method.

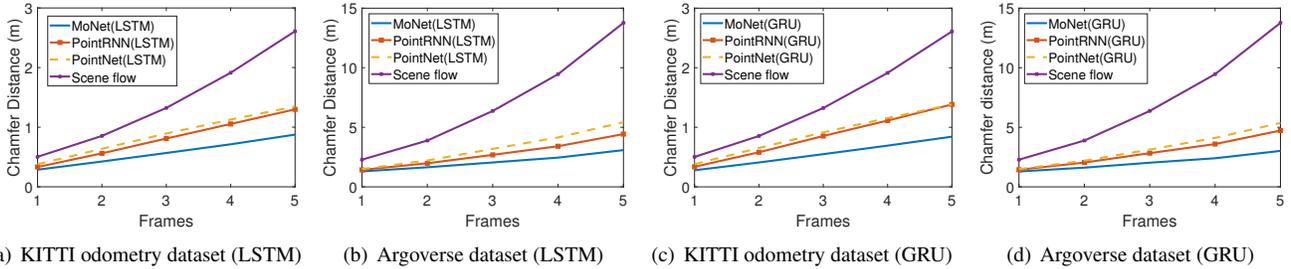


Figure 5. Chamfer Distance of the proposed methods and baseline methods on KITTI odometry dataset and Argoverse dataset. The left two images display the results using LSTM and the right two images show the results using GRU.

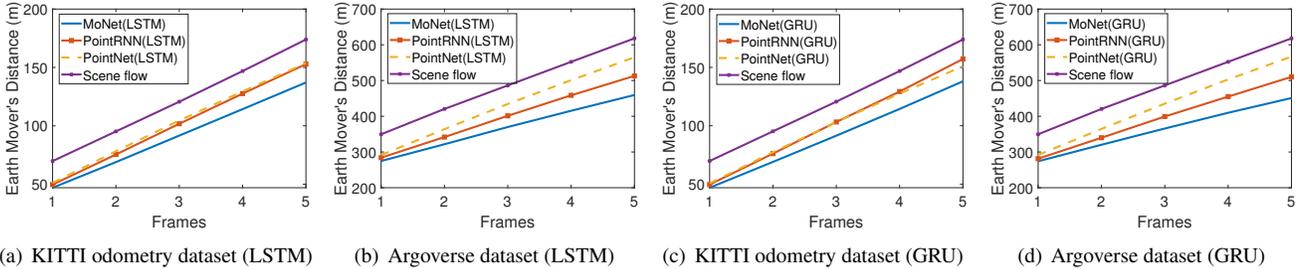


Figure 6. Earth Mover's Distance (EMD) of the proposed methods and baseline methods on KITTI odometry dataset and Argoverse dataset. The left two images display the EMD using LSTM and the right two images show the results using GRU.

Table 1. Computation time (ms) for predicting 5 frames with different number of points.

Number of points	16384	32768	65536
MoNet (GRU)	91	129	280
PointRNN (GRU)	89	125.5	268.7
PointNet++ (GRU)	48	86	233
Scene flow	64	115	216

Table 2. Precision, average Euclidean distance (AED) and average heading error (AHE) of 3D object detection results from predicted point clouds of different methods.

Method	Precision	AED (m)	AHE ( $^{\circ}$ )
MoNet (GRU)	<b>0.763</b>	<b>0.173</b>	<b>2.267</b>
PointRNN (GRU)	0.498	0.232	2.862
PointNet++ (GRU)	0.458	0.237	3.039
Scene flow	0.444	0.231	2.887

#### 4.5.3 Earth Mover's Distance

Similar to Chamfer Distance, Earth Mover's Distance (EMD) on two datasets are shown in Fig. 6. Compared with CD, EMD is more sensitive to the local details and the density distribution of the point clouds [16]. According to Fig. 6, the proposed MoNet significantly outperforms other baseline methods on this metric. For example, the EMD of the 5th frame on Argoverse dataset of our MoNet (GRU) is about 8% smaller than PointRNN (GRU) and 15% smaller than PointNet++ (GRU). The lower EMD indicates that our methods preserve the local details of point clouds better, which is also verified by the qualitative experiments results.

#### 4.6. Efficiency

Noting that the content features and motion features of previous frames do not need to be recalculated when a new point cloud is generated by the LiDAR. The computation time for predicting 5 future frames with 16384, 32768 and 65536 points is shown in Table 1. According to the results, the proposed method has a similar computation time with PointRNN, however, achieves much better performance. Specifically, the computation time of our method is 280 ms even with 65536 points, which is lower than the time required for typically LiDAR (10 Hz) to generate 5 frames.

#### 4.7. Applications

The predicted point clouds can be applied to many applications like 3D object detection and semantic segmentation. Taken 3D object detection as an example, we adopt PV-RCNN [28] to detect cars from the original point clouds and the predicted ones. We use a metric named precision to evaluate the quality of the predicted point clouds. A car is detected if the predicted score is larger than a threshold  $\tau_s = 0.9$ . For each detected car in the original point cloud, we search for the nearest detection in the predicted point clouds and if the 2D Euclidean distance is within a threshold  $\tau_d = 0.5m$  and the heading error is within  $\tau_{\theta} = 10^{\circ}$ , the detection in the predicted point clouds is considered as valid and the precision is defined as the ratio of the valid detections. Besides, we also calculate the average Euclidean distance (AED) and average heading error (AHE) of valid detections to evaluate the accuracy of the predicted point

Table 3. Average CD (m) and EMD (m) of MoNet (LSTM) and MoNet (GRU) on two datasets.

Dataset	Model	CD	EMD
KITTI	MoNet (LSTM)	0.573	<b>91.79</b>
	MoNet (GRU)	<b>0.554</b>	91.97
Argoverse	MoNet (LSTM)	2.105	368.21
	MoNet (GRU)	<b>2.069</b>	<b>364.14</b>

clouds. The experiment is performed on Sequence 08 of KITTI dataset and the number of point clouds is set to 65536. The results are shown in Table 2. According to Table 2, the average precision across 5 frames of the proposed method is 0.763, which outperforms other baseline methods by a significant margin and the results also demonstrate the high consistency between the predicted point clouds and the original ones. Besides, the low AED and AHE show that the valid detections in the predicted point clouds are close to the original ones. Visualizations of 3D object detection are displayed in the supplementary material. The experiments on 3D object detection show that the predicted point clouds can be used for further perception and indicate the great application potential of the proposed method.

## 4.8. Ablation study

### 4.8.1 MotionLSTM or MotionGRU?

Noting that we provide two versions of MotionRNN (*i.e.*, MotionLSTM and MotionGRU) to model the temporal correlations. To compare the performance of the two versions, we calculate the average Chamfer Distance (CD) and Earth Mover’s Distance (EMD) of 5 future point clouds of MoNet (LSTM) and MoNet (GRU) and display the results in Table 3. According to the results, the performance of MoNet (LSTM) and MoNet (GRU) is highly similar, which indicates that the proposed method is not sensitive to different recurrent neural network architectures. The performance of MoNet (GRU) is slightly better than MoNet (LSTM). Besides, the network parameters of MotionGRU is less than that of MotionLSTM, which also results in a faster inference speed and lower memory usage. Overall, MoNet (GRU) can be a better choice for point cloud prediction.

### 4.8.2 Motion features and content features

As we claimed before, motion features and content features can both contribute to the prediction of future point clouds. To demonstrate the significance of the combination of motion and content features, we remove the motion features and content features separately to compare the performance. The rows from top to bottom of Fig. 7 display the point clouds of a vehicle from ground truth point cloud, the results of full MoNet (GRU) and the results of

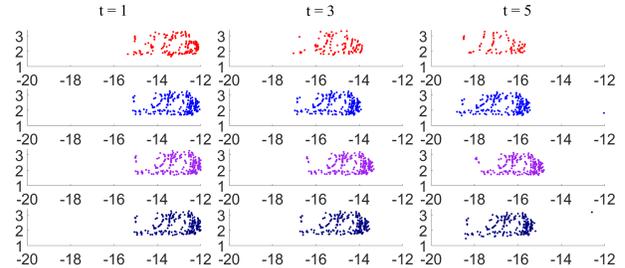


Figure 7. From left to right are future frames  $t = 1, 3, 5$ . From top to bottom are results of ground truth, full MoNet (GRU), model w/o motion features and w/o content features.

Table 4. Average CD (m) and EMD (m) of MoNet (GRU), without (w/o) motion features and w/o content features on two datasets.

Dataset	Options	CD	EMD
KITTI	full model	<b>0.554</b>	<b>91.97</b>
	w/o motion features	0.646	92.98
	w/o content features	0.637	93.83
Argoverse	full model	<b>2.069</b>	<b>364.14</b>
	w/o motion features	2.200	372.48
	w/o content features	2.158	366.40

the model without motion and content features. The model without motion features results in biased motion estimation. The model without content features can correctly predict the motion, however, the point clouds are slightly deformed. The average CD and EMD of MoNet (GRU) with and without motion features and content features are shown in Table 4. According to the results, the combination of content and motion features significantly improves the performance of the proposed MoNet. For example, the CD without motion features and content features are 0.092 m and 0.083 m higher than full MoNet (GRU) on KITTI dataset, respectively. Besides, the EMD of the full model is also lower than the model without motion or content features. Overall, the combination of content and motion features leverages the strength of both features to obtain precise motion estimation and better preservation of local details.

## 5. Conclusion

In this paper, we explore a problem named Point Cloud Prediction, which aims to predict future frames given past point cloud sequence. To achieve that, we propose a novel motion-based neural network named MoNet. Specifically, the proposed MoNet integrates motion features into the prediction pipeline and combines that with content features. Besides, a recurrent neural network named MotionRNN is proposed to capture the temporal correlations of both features across point cloud sequence and a novel motion align module is adopted to estimate motion features without future point cloud frames. Both qualitative and quantitative

experiments are performed on KITTI odometry dataset and Argoverse dataset to demonstrate the performance of the proposed method. Abundant ablation studies show that the combination of motion and content features enables the model to precisely predict the motions and also well preserve the structures. Moreover, experiments on applications reveal the practical potential of the proposed method.

## References

- [1] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 8748–8757, 2019. 2, 5
- [2] Xinyuan Chen, Chang Xu, Xiaokang Yang, and Dacheng Tao. Long-term video prediction via criticization and retrospection. *IEEE Transactions on Image Processing*, 29:7090–7103, 2020. 1
- [3] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014. 1
- [4] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3075–3084, 2019. 2
- [5] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 605–613, 2017. 5, 6
- [6] Hehe Fan and Yi Yang. Pointtrnn: Point recurrent neural network for moving point cloud processing. *arXiv preprint arXiv:1910.08287*, 2019. 1, 2, 4, 5
- [7] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3354–3361. IEEE, 2012. 2, 5
- [8] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3d point clouds: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 2020. 1
- [9] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 1
- [10] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. Randla-net: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020. 1
- [11] Beibei Jin, Yu Hu, Qiankun Tang, Jingyu Niu, Zhiping Shi, Yinhe Han, and Xiaowei Li. Exploring spatial-temporal multi-frequency analysis for high-fidelity and temporal-consistency video prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4554–4563, 2020. 1
- [12] Nal Kalchbrenner, Aaron Oord, Karen Simonyan, Ivo Danihelka, Oriol Vinyals, Alex Graves, and Koray Kavukcuoglu. Video pixel networks. In *International Conference on Machine Learning*, pages 1771–1779. PMLR, 2017. 2
- [13] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations*, 2015. 5
- [14] Yong-Hoon Kwon and Min-Gyu Park. Predicting future frames using retrospective cycle gan. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1811–1820, 2019. 1, 2
- [15] Xiaodan Liang, Lisa Lee, Wei Dai, and Eric P Xing. Dual motion gan for future-flow embedded video prediction. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1744–1752, 2017. 2
- [16] Minghua Liu, Lu Sheng, Sheng Yang, Jing Shao, and Shi-Min Hu. Morphing and sampling network for dense point cloud completion. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 11596–11603, 2020. 7
- [17] Xingyu Liu, Charles R Qi, and Leonidas J Guibas. Flownet3d: Learning scene flow in 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 529–537, 2019. 2, 5
- [18] Xingyu Liu, Mengyuan Yan, and Jeannette Bohg. Meteor-net: Deep learning on dynamic 3d point cloud sequences. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 9246–9255, 2019. 2
- [19] Andres Milioto, Ignacio Vizzo, Jens Behley, and Cyrill Stachniss. Rangenet++: Fast and accurate lidar semantic segmentation. In *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4213–4220. IEEE, 2019. 1
- [20] Yuecong Min, Yanxiao Zhang, Xiujuan Chai, and Xilin Chen. An efficient pointlstm for point clouds based gesture recognition. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5761–5770, 2020. 2
- [21] Sajjad Mozaffari, Omar Y Al-Jarrah, Mehrdad Dianati, Paul Jennings, and Alexandros Mouzakitis. Deep learning-based vehicle behavior prediction for autonomous driving applications: A review. *IEEE Transactions on Intelligent Transportation Systems*, 2020. 1
- [22] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimeshain, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037, 2019. 5
- [23] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 652–660, 2017. 1
- [24] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on

- point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 3, 4, 5
- [25] Yossi Rubner, Carlo Tomasi, and Leonidas J Guibas. The earth mover’s distance as a metric for image retrieval. *International journal of computer vision*, 40(2):99–121, 2000. 6
- [26] Andrey Rudenko, Luigi Palmieri, Michael Herman, Kris M Kitani, Dariu M Gavrila, and Kai O Arras. Human motion trajectory prediction: A survey. *The International Journal of Robotics Research*, 39(8):895–935, 2020. 1
- [27] Hanyu Shi, Guosheng Lin, Hao Wang, Tzu-Yi Hung, and Zhenhua Wang. Spsequencenet: Semantic segmentation network on 4d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4574–4583, 2020. 2
- [28] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10529–10538, 2020. 1, 7
- [29] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. Point-rcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–779, 2019. 1
- [30] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015. 4
- [31] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *International Conference on Learning Representations*, 2017. 2
- [32] Yunbo Wang, Lu Jiang, Ming-Hsuan Yang, Li-Jia Li, Mingsheng Long, and Li Fei-Fei. Eidetic 3d lstm: A model for video prediction and beyond. In *International Conference on Learning Representations*, 2018. 1, 2
- [33] Yunbo Wang, Mingsheng Long, Jianmin Wang, Zhifeng Gao, and S Yu Philip. Predrnn: Recurrent neural networks for predictive learning using spatiotemporal lstms. In *Advances in Neural Information Processing Systems*, pages 879–888, 2017. 1, 2
- [34] Xinshuo Weng, Jianren Wang, Sergey Levine, Kris Kitani, and Rhinehart Nick. Inverting the Pose Forecasting Pipeline with SPF2: Sequential Pointcloud Forecasting for Sequential Pose Forecasting. *CoRL*, 2020. 1, 2
- [35] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. Squeezeseg: Convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1887–1893. IEEE, 2018. 1
- [36] SHI Xingjian, Zhouong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015. 2, 4
- [37] Yan Yan, Yuxing Mao, and Bo Li. Second: Sparsely embedded convolutional detection. *Sensors*, 18(10):3337, 2018. 1
- [38] Zetong Yang, Yanan Sun, Shu Liu, and Jiaya Jia. 3dssd: Point-based 3d single stage object detector. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11040–11048, 2020. 1
- [39] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. Polarnet: An improved grid representation for online lidar point clouds semantic segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9601–9610, 2020. 1
- [40] Zhishuai Zhang, Jiyang Gao, Junhua Mao, Yukai Liu, Dragomir Anguelov, and Congcong Li. Stinet: Spatiotemporal-interactive network for pedestrian detection and trajectory prediction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11346–11355, 2020. 1