2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

ATPPNet: Attention based Temporal Point cloud Prediction Network

Kaustab Pal*[1], Aditya Sharma*[1], Avinash Sharma[2], K. Madhava Krishna[1]

***Abstract*— Point cloud prediction is an important yet challenging task in the field of autonomous driving. The goal is to predict future point cloud sequences that maintain object structures while accurately representing their temporal motion. These predicted point clouds help in other subsequent tasks like object trajectory estimation for collision avoidance or estimating locations with the least odometry drift. In this work, we present ATPPNet, a novel architecture that predicts future point cloud sequences given a sequence of previous time step point clouds obtained with LiDAR sensor. ATPPNet leverages Conv-LSTM along with channel-wise and spatial attention dually complemented by a 3*D*-CNN branch for extracting an enhanced spatio-temporal context to recover high quality fidel predictions of future point clouds. We conduct extensive experiments on publicly available datasets and report impressive performance outperforming the existing methods. We also conduct a thorough ablative study of the proposed architecture and provide an application study that highlights the potential of our model for tasks like odometry estimation.**

I. Introduction

Autonomous navigation is a widely explored research direction in the robotics domain with applications in autonomous aerial/aquatic drones, vehicles, mobile robots, etc. Recent advancements in 3*D* sensing led by commercial LiDAR (Light Detection and Ranging) technology have reinvigorated interest in this field as LiDAR sensors yield large-scale real-time sequential point clouds (also represented as range images), providing high-fidelity perception of the 3*D* world in comparison to traditional monocular/stereo based vision solutions. The availability of such large-scale data [1], [2] has enabled researchers to explore relevant complex tasks such as Localization [3], [4], Place Recognition [5], Segmentation [6], [7] and Obstacle Trajectory Prediction [8]. The majority of existing methods attempting to solve these tasks rely on captured sequential point clouds available in a given temporal window of the recent past. Interestingly, predicting the future 3*D* point cloud that the sensor is likely to see, can immensely enhance the performance of autonomous navigation tasks like active localization [9].

Nevertheless, the task of predicting future point clouds comes with its own set of challenges. One key challenge is that the point clouds are unordered in the space dimension (albeit ordered temporally) and vary in sampling size hence it is difficult to model spatio-temporal coherence among them. As a result, conventional architectures for feature encoding (e.g., CNNs) and sequence prediction (e.g., LSTMs) cannot be directly employed as they cannot process spatially unordered data. Another key challenge is that the LiDAR point clouds are extremely sparse making it difficult to capture the geometrical structures of the objects in the scene and hence predicting them in the future timesteps is extremely difficult. The noise in sensing puts additional challenges in the perception of real-world scenes where objects are largely cluttered. Moreover, each full-scale point cloud contains more than $100,000$ points. Extracting features from these sequences of full-scale point clouds becomes a memory-intensive task.

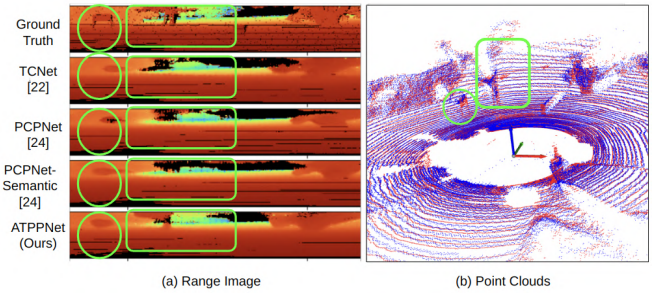

Fig. 1: (a) Predicted range images by our ATPPNet and existing methods in comparison to ground truth and, (b) the 3*D* rendering of the predicted point cloud by ATPPNet (blue) and ground-truth (red). Green circle/rectangle highlights regions where ATPPNet's predictions are superior.

Traditionally, 3*D* data is processed with deep learning encoders using volumetric [10]–[12], point-cloud [13], [14] and multi-view projection [15]–[17] methods. In regard to future point cloud prediction, primarily two lines of work exist, focusing on point cloud and range image representation. The existing point cloud prediction methods either reformulate the task as scene flow estimation [18] or employ RNN kind of temporal prediction [19], [20]. The former predicts just a translation of the 3*D* points and hence does not represent the future point cloud accurately. At the same time, the latter works on down-sampled point clouds (for memory efficiency reasons) thereby limiting the resolution of 3*D* data.

On the other hand, range image based representations project the point cloud data to a 2*D* virtual image plane of the LiDAR sensor, thereby retaining only the single (closest, farthest, or average) depth of the scene for every pixel. Early work with this representation [21] used LSTMs to process the temporal sequences and predict a sequence of future range images. [22] used 3*D*-CNNs with circular padding and skip-connections to predict a sequence of future range images while [23] used Conv-LSTMs on each of the features from the convolution encoder for the prediction task. However, their network is cumbersome and they use the auto-regressive approach for prediction of range images. Recent work in [24]

* denotes equal contribution.
[1] are with RRC, IIIT Hyderabad, India. {kaustab21, meduri99aditya}@gmail.com, mkrishna@iiit.ac.in
[2] is with IIT Jodhpur, India. avinashsharma@iitj.ac.in
Codebase is available at https://tinyurl.com/atppnet

uses the self-attention mechanism of Transformers along with a semantic-based loss function. This method compresses the $3D$ tensors into height and width dimensions and processes each of them separately using two separate transformer blocks. As a result, they are using self-attention only on the channels and since they are compressing the feature tensor into height and width they are also losing the spatial context. Additionally, their model size in terms of the number of parameters is large.

In this paper, we propose a novel architecture for predicting future point clouds from a given sequence of past point clouds represented as LiDAR range images. More specifically, we propose *ATPPNet: Attention based Temporal Point cloud Prediction Network* that leverages Conv-LSTM [25] blocks along with channel-wise and spatial attention modules for extracting an enhanced spatio-temporal context for the task of future point cloud prediction. Further, we also leverage a complimentary $3D$-CNN branch to spatio-temporally encode the global feature embeddings of the range images. Additionally, we also predict the re-projection mask associated with the predicted range images to retain only the valid range values when re-projecting to the point cloud. Compared to [24], we show that processing the range image sequences using Conv-LSTM and using spatial and channel-wise attention directly on learned spatio-temporal $3D$ features works better without the need for a separate semantic-based loss function. Our proposed architecture achieves state-of-the-art performance on two publicly available datasets. Our method yields real-time future point cloud prediction (faster than a typical rotating $3D$ LiDAR sensor point cloud rate i.e., 10Hz). We conduct thorough qualitative and quantitative evaluations as well as provide a detailed ablation study to validate the effectiveness of our proposed architecture. To summarize, our main contributions are as follows:

- We proposed a novel architecture (ATPPNet) that leverages Conv-LSTM and spatial and channel-wise attention for predicting future point clouds from a sequence of past point clouds.
- ATPPNet achieves SOTA performance on various publicly available datasets while beating the existing methods by $8-9\%$ margin.
- We empirically show that ATPPNet improves the performance of downstream tasks, like odometry estimation.

II. Our Approach

We provide details of our novel ATPPNet (Attention based Temporal Point cloud Prediction Network) that leverages Conv-LSTM blocks along with channel-wise and spatial attention modules complemented by a 3D-CNN branch to process a past sequence of $3D$ point clouds to predict future point clouds. Let $S_P = \{S_{t-M+1}, S_{t-M+2}, \dots S_t\}$ be the set of input $3D$ point cloud sequence of M time steps in a temporal window where $S_\tau \in \mathbb{R}^3$ is set of $3D$ points captured at specific time step τ. The goal of our ATPPNet is to predict the future sequence of $3D$ point clouds for a temporal window of N time steps represented as $S_F = \{S_{t+1}, S_{t+2}, \dots, S_{t+N}\}$. Similar to [22], we adopt the range image representation by first converting the point clouds into the spherical coordinate system and then projecting the corresponding point cloud ($S_\tau \in \mathbb{R}^3$) to the virtual image plane of the LiDAR sensor, represented as $R_\tau \in \mathbb{R}^2$. Let $R_P = \{R_{t-M+1}, R_{t-M+2}, \dots R_t\}$ be the sequence of past range images in a fixed temporal window (obtained from S_P) and similarly $R_F = \{R_{t+1}, R_{t+2}, \dots, R_{t+N}\}$ be the sequence of predicted range images associated with S_F.

A. Overall Architecture

Figure 2 provides the overview of the proposed ATPPNet architecture. A shared convolution encoder processes the input range images R_P and generates L number of multi-scale feature tensors for each of the range images. Subsequently, the first $L-1$ feature tensors are fed to Conv-LSTMs to model the spatio-temporal relationships across R_P. Further, we exploit spatial as well as channel-wise attention on the outputs of Conv-LSTMs to obtain the $L-1$ *context tensors* (i.e., the consolidated spatio-temporal encoding of R_P). Additionally, for the final L-th feature tensor, we use a $3D$-CNN layer to process the spatio-temporal relationship and generate the L-th feature tensor for N future time steps. On the decoder side, we feed the context tensor along with the hidden state of the last time step to $L-1$ Conv-LSTMs and generate the feature tensors for each of the $L-1$ layers for N time steps into the future. All these L feature tensors on the decoder side are subsequently processed to generate the range image sequence R_F along with their corresponding re-projection masks M_F for all the N future time steps where each pixel of $M_\tau \in M_F$ can be interpreted as the probability for each of the range image pixels to be valid or invalid. This re-projection mask is used while back-projecting a range image to a point cloud where we only retain the range values corresponding to probabilities greater than 0.5. The construction of specific architectural blocks is given below.

B. Convolution Encoder & Decoder block

The convolution encoder block takes the range image and first performs a $2D$-convolution operation, resulting in a tensor with an increased number of channels but the same spatial resolution. This tensor is further processed using L convolutional sub-blocks. Each sub-block takes as input a feature tensor and applies a combination of $2D$-convolution, $2D$-batch normalization, and leaky-ReLU operation while keeping the tensor dimensions the same. A strided-convolution operation is subsequently performed, resulting in a down-sampled tensor.

The convolution decoder block follows the reversed structure of the convolution encoder block. There are L sub-blocks, each of which takes an input tensor and passes it through a $2D$-transposed convolution, $2D$-batch normalization, and leaky-ReLU operation resulting in a spatially scaled-up tensor while keeping the number of channels the same. Another $2D$-convolution operation is then performed, to decrease the channel size of the tensor. The output of the L-th layer is finally passed through another $2D$-convolutional

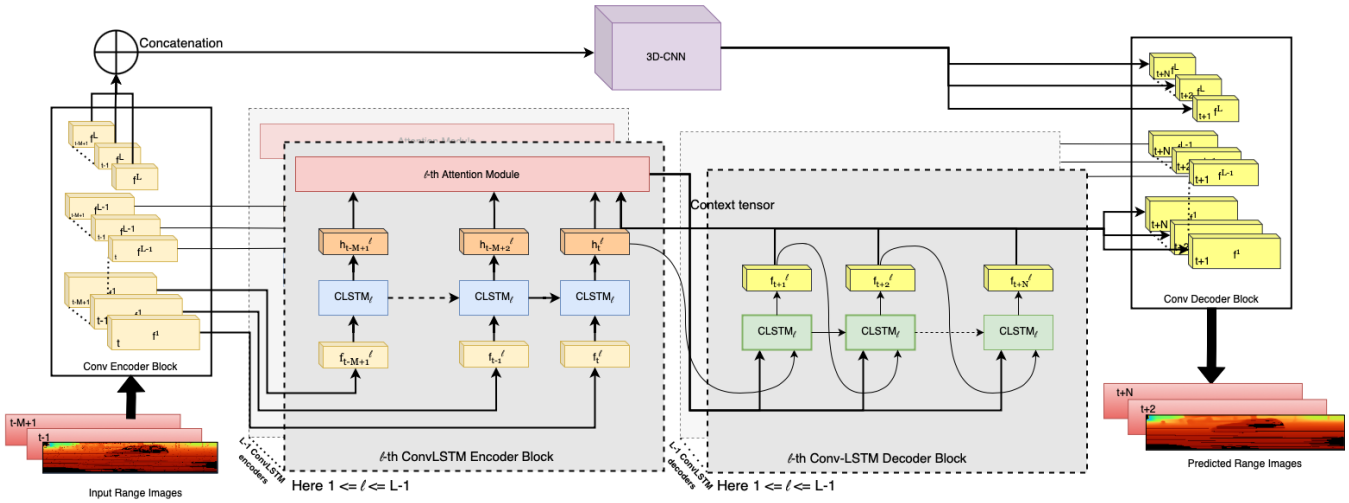

Fig. 2: **ATPPNet Architecture.** ATPPNet leverages Conv-LSTM along with channel-wise and spatial attention dually complemented by a 3D-CNN branch for extracting an enhanced spatio-temporal context to recover high quality fidel predictions of future point clouds.

layer resulting in the predicted range images and the associated re-projection mask.

C. Conv-LSTM encoder

We propose to use $L-1$ Conv-LSTMs to exploit the spatio-temporal context of input sequences. Similar to the S2Net [23] architecture, we used multiple Conv-LSTM layers for each of the feature tensors. This helps us in preserving the high-frequency details across the range image sequences. The Conv-LSTMs for each of the $L-1$ features generate a hidden state for each of the M time steps.

D. Attention Module

Let $\tau \in [t-M+1,\ldots,t-1,t]$ be a specific time step in the given input temporal window, and the feature tensors for the layer l at every time step be represented as f_τ^l. Similarly, let the output of l-th Conv-LSTM at time step t denoted as g_t^l where $l \in [1,\ldots L-1]$. As part of the attention module, we first compute the joint embedding $\mathscr{J}_\tau^l$ of f_τ^l and g_t^l as (using the formulation [26]):

$$\mathscr{J}_\tau^l = \sigma_1(W_f f_\tau^l \oplus W_g g_t^l),$$

where σ_1 is a non-linear activation function chosen as ReLU and $\oplus$ is the concatenation operation. W_f and W_g are implemented as $2D$-convolution operations with 1×1 kernel. The use of σ_1, W_f, and W_g allows the network to learn non-linear relationships between the features, which is especially important when the image is noisy like our range images. The resulting tensors $\mathscr{J}_\tau^l$ are passed through the spatial and channel-wise attention module [27] to find the $3D$ attention map $\mathscr{M}(\mathscr{J}_\tau^l) \in R^{C_l\times H_l\times W_l}$. The refined feature tensor for layer l at time step τ is computed as:

$$\hat{f}_\tau^l = f_\tau^l \otimes \mathscr{M}(\mathscr{J}_\tau^l).$$

Here $\otimes$ is the element-wise multiplication. To compute the $3D$ attention map $\mathscr{M}(J_\tau^l)$, we compute the channel-wise attention and spatial attention separately and then combine them as

$$\mathscr{M}(\mathscr{J}_\tau^l) = \sigma(M_c(\mathscr{J}_\tau^l) \otimes M_s(\mathscr{J}_\tau^l))a,$$

where σ is the Sigmoid function and $\otimes$ is the element-wise multiplication operation. The M_c function first applies the global average pooling operation on the $3D$ tensor $\mathscr{J}_\tau^l$ to get the channel tensor which is then passed through an MLP layer to get the channel-wise attention values. The M_s function applies $2D$-convolution operation on the $3D$ tensor $\mathscr{J}_\tau^l$ and returns a single channel tensor which represents the spatial attention values.

The refined feature tensors for all the M time steps for each of the $L-1$ layers are then used to compute the context tensors, that are subsequently served as input to the decoder Conv-LSTMs.

$$context_l^t = \sum_{\tau=t}^{t-M+1} \hat{f}_\tau^l$$

E. Conv-LSTM decoder

The Conv-LSTM decoder follows a similar structure as the Conv-LSTM encoder. $L-1$ Conv-LSTM decoders are used to predict $L-1$ feature tensors for each of the N future time steps. Let $\tau \in [t+1,\ldots,N]$ be a specific time step in the predicted future temporal window. For the τ-th time step, the l-th Conv-LSTM decoder takes as input the context tensor $context_l^{\tau-1}$ where $l \in [1,\ldots,L-1]$ along with the hidden state of the Conv-LSTM for time step $\tau-1$ to compute the output feature tensor. This output feature tensor along with the hidden states of the previous time steps are used to re-compute the context tensor $context_l^\tau$ to be used for the next time step. These output feature tensors on the Conv-LSTM decoder side are used by the convolutional decoder to generate the predicted range images R_F

F. 3D-CNN block

The feature tensors for the L^{th} layer from the convolutional encoder block for all the past M time steps are concatenated to create a $4D$ tensor. A $3D$-CNN layer is used to process

this feature tensor and generate N feature maps for the L-th convolutional decoder block.

It is important to note that since $3D$-CNNs place their spatial focus on fewer, contiguous areas in the feature tensors [28], we employ $3D$-CNN on the last $L-th$ layer of the feature tensor (obtained with $2D$-CNN) as it tends to capture global structures in the range image [29], [30]. Thus, the $3D$-CNN block extracts only the complementary spatio-temporal context as the primary spatio-temporal context is already obtained by applying Conv-LSTM's on the initial layers of the convolutional encodings as they tend to capture the high frequency details in the range images. This also gives us the additional advantage of speeding up our inference time.

G. Loss Function

We use a combination of losses when training the network. Since our ground truth point clouds are projected onto $2D$ range images of dimension $H \times W$, we can use $2D$ image-based losses.

Firstly, we use the average range loss $\mathscr{L}_R$ to compute the error between the predicted range values $\hat{r}_{\tau,i,j} \in \mathbb{R}^{N\times H\times W}$ and the ground-truth range values $r_{\tau,i,j} \in \mathbb{R}^{N\times H\times W}$. The average range loss can be formulated as:

$$\mathscr{L}_R = \frac{1}{N\times H\times W}\sum_{\tau=t+1}^{t+N}\sum_{i=1}^{H}\sum_{j=1}^{W} || \hat{r}_{\tau,i,j} - r_{\tau,i,j} ||_1,$$

where $||\bullet||_1$ represents the L_1 norm. The range loss $\mathscr{L}_R$ is computed only for the valid ground truth points. To train the re-projection mask output, we use the Binary Cross-Entropy loss between the predicted mask values $\hat{m}_{\tau,i,j} \in \mathbb{R}^{N\times H\times W}$ and the ground truth mask values $m_{\tau,i,j} \in \mathbb{R}^{N\times H\times W}$. The average mask loss $\mathscr{L}_M$ is computed as:

$$\mathscr{L}_M = \frac{1}{N\times H\times W}\sum_{\tau=t+1}^{t+N}\sum_{i=1}^{H}\sum_{j=1}^{W} -m_{\tau,i,j}\log\hat{m}_{\tau,i,j} \quad (1)$$

$$-(1-m_{\tau,i,j})\log(1-\hat{m}_{\tau,i,j}), \quad (2)$$

where $\hat{m}_{\tau,i,j}$ is the predicted probability of whether the range value is valid. $m_{\tau,i,j}$ is 1 if the ground-truth range value is valid and 0 otherwise. A masked range image is generated by taking only the range values from the range image whose corresponding mask values are greater than 0.5. Since we are re-projecting the predicted masked range images into point clouds, we use Chamfer distance [31] represented as $\mathscr{L}_C$ for evaluating fidelity of the predicted point clouds.

The combined loss function is given as

$$\mathscr{L} = \mathscr{L}_R + \mathscr{L}_M + \alpha_C\mathscr{L}_C.$$

α_C is the weight associated with the Chamfer distance.

III. Experiments and Results

A. Experimental Settings

We train ATPPNet in a self-supervised manner in the sense that we use only sequential point cloud data sans no manually annotated labels. For our experiments, we keep the temporal window size $M = N = 5$. In the convolutional encoder block, the initial convolutional operation outputs 16 channels while retaining the spatial dimension. We use $L = 4$ sub-blocks where the channel size increases by a factor of 2 for every successive sub-block obtained by the convolutional encoder. In each of the sub-blocks, the first convolutional operation uses a kernel size of 3×3 with stride $(1,1)$, and the second convolution operation uses a kernel size of 2×4 with stride $(2,4)$. All the convolutional operations use circular padding [22]. Each Conv-LSTM block uses 3 layers.

Prediction Step	TCNet [22]	PCPNet [24]	PCPNet-Semantic [24]	ATPPNet (Ours)
1	0.554	0.543	0.503	**0.468**
2	0.671	0.662	0.620	**0.570**
3	0.779	0.773	0.727	**0.667**
4	0.878	0.872	0.825	**0.760**
5	0.974	0.973	0.920	**0.851**
Mean	0.771	0.765	0.719	**0.663**

TABLE I: Range Loss results on the KITTI odometry test set verifies that ATPPNet has a performance improvement of 7.788% over SOTA on the mean range loss. Bold values correspond to the best performing model in that corresponding time step.

Similar to the trend in the literature [22], [24], we train our architecture for 50 epochs with $\alpha_C = 0$ and then fine-tuned with the Chamfer distance loss for the next 10 epochs by setting $\alpha_C = 1$. We train our model on a system with an Intel Xeon E5-2640 CPU and 3 Nvidia RTX 2080 GPUs using the Distributed Data Parallel strategy. While training, we have used the ADAM optimizer [32] with default parameters and an initial learning rate of 0.0003 and the StepLR learning rate scheduler with gamma as 0.99.

1) KITTI Odometry dataset [2]: We use sequences $00-05$ for training, $06-07$ for validation, and $08-10$ for testing. The LiDAR used in the KITTI dataset [2] has 64 channels, so we have used range images of size 64×2048.

2) nuScenes dataset [1]: We trained our network on this dataset with the same training strategy we used on the KITTI dataset. Following PCPNet [24], we used sequence $00-69$ for training, scenes $70-84$ for validation, and $85-99$ for testing. We trained our architecture on range images of size 32×1024 since the LiDAR used here has 32 channels.

B. Qualitative Analysis

Figure 3 (and Figure 1) shows a qualitative comparison of the predicted point clouds generated using our proposed ATPPNet and the other methods: TCNet [22], PCPNet and PCPNet-semantic [24]. The areas of interest are highlighted with numbered green circles.

In the predicted sequence $t+1$ shown in Figure 3, we can observe over the circles a,b,c & d that our ATPPNet outperforms the other methods by generating point clouds that are less noisy and structurally more similar to the ground truth. We can observe in time step $t+5$ (bottom row) that the circles numbered a,b,c,d,e & f in the point cloud generated by ATPPNet is more fidel to the ground truth compared to the predicted point clouds from the other methods that have large visible deviations from the ground truth and are more noisy.

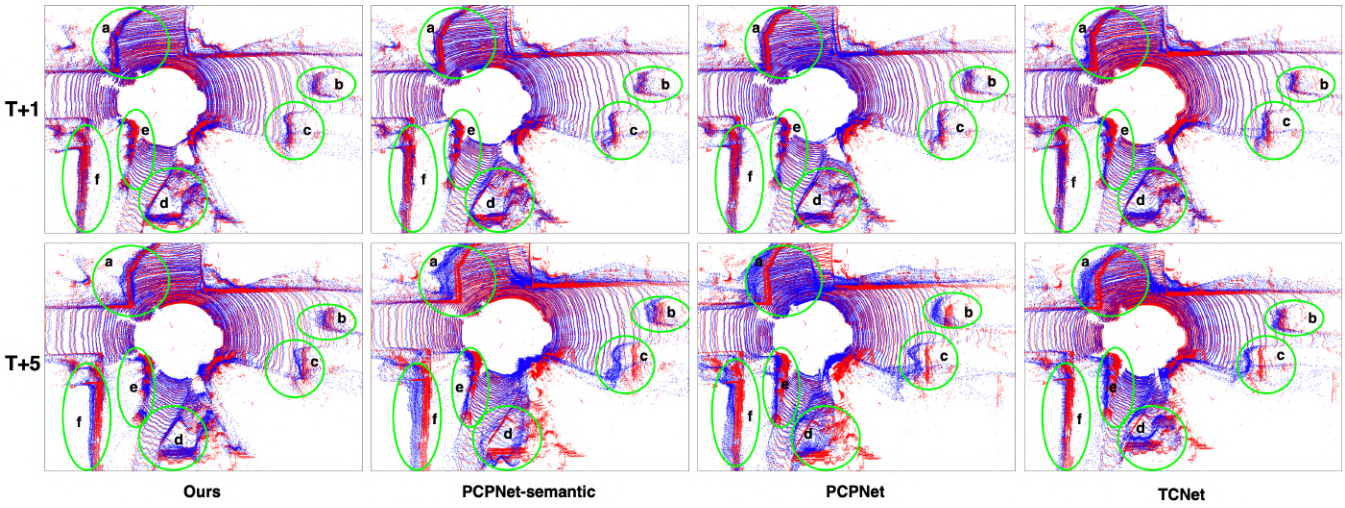

Fig. 3: Qualitative comparison conducted on sequence 10 of the KITTI odometry dataset. The predicted points (blue) and the ground truth points (red) are combined for a better visual comparison. The top row shows the point clouds at prediction step $t+1$ and the bottom row shows the point clouds at prediction step $t+5$. The areas of interest are circled in green.

	Sampled Point Cloud						**Full-Scale Point Clouds**			
Prediction Step	PointLSTM [19]	MoNet [20]	TCNet [22]	PCPNet [24]	PCPNet-Semantic [24]	ATPPNet (Ours)	TCNet [22]	PCPNet [24]	PCPNet-Semantic [24]	ATPPNet (Ours)
1	0.332	0.278	0.290	0.285	0.280	**0.258**	0.253	0.252	0.242	**0.225**
2	0.561	0.409	0.357	0.341	0.340	**0.311**	0.309	0.301	0.298	**0.270**
3	0.810	0.549	0.441	0.411	0.412	**0.375**	0.377	0.362	0.354	**0.326**
4	1.054	0.692	0.522	0.492	0.495	**0.445**	0.448	0.435	0.427	**0.391**
5	1.299	0.842	0.629	0.580	0.601	**0.523**	0.547	0.514	0.503	**0.461**
Mean	0.811	0.554	0.448	0.422	0.426	**0.382**	0.387	0.373	0.365	**0.335**

TABLE II: Chamfer distance results on KITTI Odometry test sequence with the sampled point clouds on the left and full-scale point clouds on the right. ATPPNet has a performance improvement of 9.478% over SOTA for the sampled point clouds and 8.219% over SOTA for the full-scale point clouds. Bold values correspond to the best performing model in that corresponding time step.

Evaluation metric	TCNet [22]	PCPNet-Semantic [24]	ATPPNet (Ours)
Mean Chamfer Distance	1.389	1.360	**0.932**
Mean Range Loss	0.719	0.704	**0.598**

TABLE III: Mean Range Loss and Chamfer distance on the nuScenes test set. ATPPNet is making an improvement of 15.056% over SOTA on the mean Range loss and 31.470% over SOTA on the mean Chamfer distance.

C. Quantitative Analysis

In this section, we perform a quantitative analysis of our proposed ATPPNet with two point based methods (PointLSTM [19], MoNet [20]) on the KITTI [2] test set, and three range image based methods (TCNet [22], PCPNet and PCPNet-semantic [24]) on the KITTI [2] and the nuScenes [1] test set. The point based methods [19], [20] use downsampled point clouds to 65536 points and this is also adopted by us and other methods i.e., [22], [24]. We use the range loss and Chamfer distance to evaluate the predicted range images and the point clouds, respectively.

Table I shows the quantitative results of the range loss for all the methods on the KITTI test set. Compared to the other methods, ATPPNet generates better range images as the prediction time step increases, which can also be verified by the improvement of 7.788% over SOTA (PCPNet-semantic [24]) in the mean range loss.

In Table II, we evaluate the Chamfer distance on the sampled point clouds (left column) and full-scale point clouds (right column) on the KITTI test set. As we can observe, our method is having an improvement of 9.478% over SOTA on sampled point clouds and an improvement of 8.219% over SOTA on full-scale point clouds. It can also be observed that the margin of Chamfer distance between ATPPNet and other methods increases as the prediction time step increases (i.e., farther in future). This indicates a more stable prediction of the point clouds across all the time steps as depicted in Figure 3.

In Table III, we report the quantitative analysis of our model trained on the nuScenes dataset. ATPPNet is improving 15.056% on the mean range loss and 31.470% on the mean Chamfer distance over SOTA. Our inference time on the KITTI and nuScenes dataset is 89.5 and 70.7 ms respectively.

D. Ablation Study

In this section, we conduct a thorough investigation of the relevance of different blocks of our architecture on the KITTI test set and demonstrate the effectiveness of our method.

Impact of Attention Module: In Table IV column *A*, we show the results of our ablation study on the attention

	A) Attention Module			B) Conv-LSTM (CLSTM) layers			
Evaluation metric	No Attention	S-Attention	C-Attention	$L-1$ CLSTM	$L-1$ & $L-2$ CLSTM	all L CLSTM	ATPPNet (Ours)
Chamfer distance	0.365	0.359	0.356	0.405	0.378	0.366	**0.335**
Range loss	0.719	0.687	0.690	0.770	0.698	0.717	**0.663**

TABLE IV: Results of Ablation study on Attention Module and Conv-LSTM layers. Bold values correspond to the best performing model.

Window size	3	5	7
T+1	**0.221**	0.255	0.258
T+2	0.274	**0.270**	0.306
T+3	0.344	**0.326**	0.363
T+4	NA	**0.391**	0.426
T+5	NA	**0.461**	0.498
T+6	NA	NA	0.580
T+7	NA	NA	0.657

TABLE V: An ablation study on how the performance changes as we vary the sequence length.

Pose error	TCNet [22]	PCPNet [24]	PCPNet-Semantic [24]	ATPPNet (Ours)
$\mathscr{L}_P^{t+1}$	0.1342	0.1363	0.1280	**0.1209**
$\mathscr{L}_P^{t+2}$	0.2412	0.2282	0.2235	**0.2065**
$\mathscr{L}_P^{t+3}$	0.3670	0.3388	0.3343	**0.3038**
$\mathscr{L}_P^{t+4}$	0.5084	0.4630	0.4558	**0.4128**
$\mathscr{L}_P^{t+5}$	0.6736	0.6022	0.5878	**0.5328**
Mean	0.3849	0.35374	0.3459	**0.3154**

TABLE VI: **LOAM pose error.** We adopt LOAM [36] and evaluate the disparity between the motion estimates on ground truth and predictions.

module. To conduct this study, we set up 3 experiments: (1) removing the attention module (column "No Attention"), (2) using just spatial attention (column "S-Attention") and (3) using just channel-wise attention (column "C-Attention"). We observe in all the 3 experiments that the range loss and Chamfer distance deteriorates as compared to our original method.

Effects of Spatio-Temporal Modelling: In Table IV column *B*, we demonstrate the impact of varying the number of feature tensors from the convolutional encoder to be modelled spatio-temporally. For this, we adopt three experimental setups: (1) modelling only the $L-1$-th feature tensor with Conv-LSTM and L-th tensor with $3D$ CNN. (column "$L-1$ CLSTM"), (2) modelling only the $L-1$-th and $L-2$-th feature tensor with Conv-LSTM and L-th tensor with $3D$ CNN. ("$L-1$ & $L-2$ CLSTM"), and (3) modelling all the L layers from the convolutional encoder with Conv-LSTM (column "all L CLSTM"). For the aforementioned experiments we observe a deterioration in the performance compared to our original method. We can also conclude the importance of the $3D$-CNN layer for the L-th layer from experiment (3). This verifies that $3D$ CNNs are better at modelling contiguous areas in the feature tensors [28] which tend to appear at the lower level features from the convolutional encoder [29], [30].

Impact of Sequence Length: In Table V, we report the results by varying the temporal window size. It is important to note that the temporal window size is kept the same for both input and output. We can observe a decrease in the Chamfer distance as we increase the window size from 3 to 5. However, further increasing the window size from 5 to 7 leads to an increase in the Chamfer distance. A possible explanation for this is that the window size of 3 is too short to model the spatio-temporal context of the scene while, for the window size of 7, the context length and the prediction horizon is too long.

E. Results on Downstream Task

In this section, we analyze the impact of our model on a downstream task of generating motion estimates (i.e., odometry) for the ego vehicle. We adopt LOAM [33], and evaluate the disparity between the motion estimates on ground truth and predictions. Let $\hat{p}^\tau \in \mathbb{R}^2$ denote the trajectory pose obtained using predicted point clouds and $p^\tau \in \mathbb{R}^2$ denote the trajectory pose obtained using ground truth point clouds at time step τ, where $\tau \in [t+1,\ldots,t+N]$. The pose error $\mathscr{L}_P^\tau$ is given as:

$$\mathscr{L}_P^\tau = || \hat{p}^\tau - p^\tau ||_2^2 .$$

As reported in Table VI, the pose error ($\mathscr{L}_P^\tau$) for ATPPNet is the least as compared to the other methods. This verifies that the improved point cloud prediction from our proposed method translates to a tangible outcome in the form of improved localization vis-a-vis other methods. Additionally, such prediction of localization error can be effectively leveraged by active localization strategies [9] that steer the vehicle to regions where the localization is expected to be better.

IV. Conclusion

In this paper, we propose a new self-supervised method for predicting future point cloud sequences from past ones. We combine spatial and channel-wise attention with Conv-LSTMs and a 3D-CNN branch to model spatio-temporal information efficiently. Our approach, ATPPNet, outperforms existing methods in experiments on various real-world datasets. We also demonstrate its potential through an application study. Future work could involve predicting point clouds by querying specific locations over time to identify areas with minimal motion drift.

V. Acknowledgement

The authors acknowledge the support provided by MeitY, Govt. of India, under the project "Capacity building for human resource development in Unmanned Aircraft System (Drone and related Technology)"

REFERENCES

[1] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[2] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE conference on computer vision and pattern recognition*. IEEE, 2012, pp. 3354–3361.

[3] X. Chen, I. Vizzo, T. Läbe, J. Behley, and C. Stachniss, "Range image-based lidar localization for autonomous vehicles," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5802–5808.

[4] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized lidar odometry and mapping on variable terrain," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2018, pp. 4758–4765.

[5] J. Ma, X. Chen, J. Xu, and G. Xiong, "Seqot: A spatial–temporal transformer network for place recognition using sequential lidar data," *IEEE Transactions on Industrial Electronics*, vol. 70, no. 8, pp. 8225–8234, 2022.

[6] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss, "Moving object segmentation in 3d lidar data: A learning-based approach exploiting sequential data," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 6529–6536, 2021.

[7] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "Rangenet++: Fast and accurate lidar semantic segmentation," in *2019 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2019, pp. 4213–4220.

[8] W. Luo, B. Yang, and R. Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2018, pp. 3569–3577.

[9] M. Omama, S. V. Sundar, S. Chinchali, A. K. Singh, and K. M. Krishna, "Drift reduced navigation with deep explainable features," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 6316–6323.

[10] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *2015 IEEE/RSJ international conference on intelligent robots and systems (IROS)*. IEEE, 2015, pp. 922–928.

[11] G. Riegler, A. Osman Ulusoy, and A. Geiger, "Octnet: Learning deep 3d representations at high resolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 3577–3586.

[12] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong, "O-cnn: Octree-based convolutional neural networks for 3d shape analysis," *ACM Transactions On Graphics (TOG)*, vol. 36, no. 4, pp. 1–11, 2017.

[13] C. R. Qi, H. Su, K. Mo, and L. 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*, 2017, pp. 652–660.

[14] X. Yan, C. Zheng, Z. Li, S. Wang, and S. Cui, "Pointasnl: Robust point clouds processing using nonlocal neural networks with adaptive sampling," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 5589–5598.

[15] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proc. ICCV*, 2015.

[16] T. Yu, J. Meng, and J. Yuan, "Multi-view harmonized bilinear network for 3d object recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 186–194.

[17] Z. Yang and L. Wang, "Learning relationships for multi-view 3d object recognition," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 7505–7514.

[18] D. Deng and A. Zakhor, "Temporal lidar frame prediction for autonomous driving," in *2020 International Conference on 3D Vision (3DV)*. IEEE, 2020, pp. 829–837.

[19] H. Fan and Y. Yang, "Pointrnn: Point recurrent neural network for moving point cloud processing," *arXiv preprint arXiv:1910.08287*, 2019.

[20] F. Lu, G. Chen, Z. Li, L. Zhang, Y. Liu, S. Qu, and A. Knoll, "Monet: Motion-based point cloud prediction network," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 8, pp. 13 794–13 804, 2021.

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

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

[23] X. Weng, J. Nan, K.-H. Lee, R. McAllister, A. Gaidon, N. Rhinehart, and K. M. Kitani, "S2net: Stochastic sequential pointcloud forecasting," in *European Conference on Computer Vision*. Springer, 2022, pp. 549–564.

[24] Z. Luo, J. Ma, Z. Zhou, and G. Xiong, "Pcpnet: An efficient and semantic-enhanced transformer network for point cloud prediction," *IEEE Robotics and Automation Letters*, vol. 8, no. 7, pp. 4267–4274, 2023.

[25] X. Shi, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Advances in neural information processing systems*, vol. 28, 2015.

[26] J. Schlemper, O. Oktay, L. Chen, J. Matthew, C. Knight, B. Kainz, B. Glocker, and D. Rueckert, "Attention-gated networks for improving ultrasound scan plane detection," *arXiv preprint arXiv:1804.05338*, 2018.

[27] J. Park, S. Woo, J.-Y. Lee, and I. S. Kweon, "Bam: Bottleneck attention module," *arXiv preprint arXiv:1807.06514*, 2018.

[28] J. Manttari, S. Broomé, J. Folkesson, and H. Kjellstrom, "Interpreting video features: A comparison of 3d convolutional networks and convolutional lstm networks," in *Proceedings of the Asian Conference on Computer Vision*, 2020.

[29] M. D. Zeiler and R. Fergus, "Visualizing and understanding convolutional networks," in *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part I 13*. Springer, 2014, pp. 818–833.

[30] J. Yosinski, J. Clune, A. Nguyen, T. Fuchs, and H. Lipson, "Understanding neural networks through deep visualization," *arXiv preprint arXiv:1506.06579*, 2015.

[31] H. Fan, H. Su, and L. 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*, 2017, pp. 605–613.

[32] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[33] 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. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.