2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

PocoNet: SLAM-oriented 3D LiDAR Point Cloud Online Compression Network

Jinhao Cui[1], Hao Zou[1], Xin Kong[1], Xuemeng Yang[1], Xiangrui Zhao[1], Yong Liu[1,*]
Wanlong Li[2], Feng Wen[2], and Hongbo Zhang[2]

***Abstract*— In this paper, we present PocoNet: Point cloud Online COmpression NETwork to address the task of SLAM-oriented compression. The aim of this task is to select a compact subset of points with high priority to maintain localization accuracy. The key insight is that points with high priority have similar geometric features in SLAM scenarios. Hence, we tackle this task as point cloud segmentation to capture complex geometric information. We calculate observation counts by matching between maps and point clouds and divide them into different priority levels. Trained by labels annotated with such observation counts, the proposed network could evaluate the point-wise priority. Experiments are conducted by integrating our compression module into an existing SLAM system to evaluate compression ratios and localization performances. Experimental results on two different datasets verify the feasibility and generalization of our approach.**

I. Introduction

Along with the development of autonomous driving systems, 3D light detection and ranging (LiDAR) sensors have shown that its wide usage in the realm of robotics, and collecting point cloud data during driving has become a key feature for many intelligent vehicles. As the raw data from LiDAR sensors, a point cloud, which is composed of points in 3D space, has also demonstrated its powerful perception ability of real-world environments. Processing, sharing and storing point clouds has recently emerged as a crucial component for robotics, for example, multi-robot simultaneous localization and mapping (MR-SLAM). This task requires point cloud data transmission within limited bandwidth from multi-robots. Besides, V2X(Vehicle to Everything) network makes remote computing and remote control possible, which also intensifies the demand for point cloud data transmitting.

However, streaming point cloud data from LiDAR sensors are a type of "big data". For example, raw point clouds generated from the Velodyne HDL-64 Sensor used in the KITTI dataset can produce over 100GB of data per hour. Moreover, loading and processing of original LiDAR point cloud data and dense maps in SLAM causes computing and storage intractability since onboard platform on vehicles has constrained resources. Hence, it is designing a SLAM-oriented method of compressing this type of data that has become an indispensable task for autonomous driving systems.

The task of SLAM-oriented point cloud compression in this paper is to select a compact subset of points to maintain localization accuracy, i.e., to find fewer 3D points that contain more poses in the whole trajectory. This task is related to the maximum converge problem, which is regarded as NP-Hard problems. Note that SLAM-oriented compression methods are few similarities as compared with previous methods of point cloud compression. To begin with, the aim of SLAM-oriented compression methods is to preserve the accuracy of localization after compression, while previous studies [1]–[8] focus on data encoding. Furthermore, real-time capability shows a significant role in SLAM-oriented compression methods. For this reason, point cloud compression in autonomous driving scenarios must support on-the-fly operating. Additionally, the desired method should be able to generalize to new environments with high efficiency.

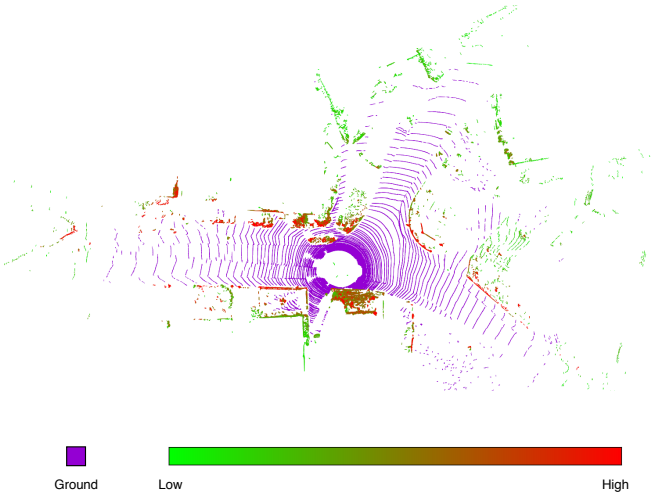


Fig. 1: Visualization of observation counts of 3D LiDAR point clouds from a raw scan on KITTI. Higher value of a observation count indicates a stronger contribution on the task of localization.

In the field of visual map compression for localization, [9]–[13] counted the observation frequency of 2D landmarks as the score function. [14] applied this idea to 3D point clouds and utilized the observation count of each point in the whole trajectory as priority levels for map compression. As shown in Fig. 1, the points with high observation counts are mostly distributed in similar parts (e.g., tree trunks, walls, telegraph poles), which can be observed frequently for vehicles on roads. As a consequence, we suppose that the observation counts associate with geometric property, which indicates that it is feasible to accomplish SLAM-oriented compression using only one frame data. Motivated by these earlier researches, we propose a compression network taking this task as point cloud segmentation. We compute observation counts of maps as scores and then assign these scores

[1]Jinhao Cui, Hao Zou, Xin Kong, Xiangrui Zhao and Yong Liu are with the Institute of Cyber-Systems and Control, Zhejiang University, Hangzhou 310027, P. R. China.(*Yong Liu is the corresponding author).
[2]The authors are with Huawei Noah's Ark Lab, Beijing, China.

to every frame of point clouds from LiDAR. A dynamic threshold setting strategy for observation counts is proposed to divide all points into different categories: high priority, low priority, and ground. Using these annotated points as supervision, we train PocoNet for point cloud segmentation to identify point-wise priority levels and then apply different selection strategies for compression. The contributions of this work can be summarized as three-fold.

- We propose PocoNet for the compression (i.e., reduction) of one frame LiDAR data. To our best knowledge, the proposed network is the first use of deep learning for SLAM oriented 3D LiDAR point cloud compression.
- Considering application scenarios, we firstly take 3D LiDAR point cloud data compression as the task of segmentation. Our network can accomplish on-the-fly operation with no need for complex handcraft features.
- Evaluation and analysis are performed to demonstrate that compressed 3D LiDAR point clouds work in SLAM and localization tasks. Besides, evaluation on the Ford Campus dataset, where we directly infer using the pre-trained model on the KITTI dataset, shows that the generalization ability of our proposed network.

II. Related Work

In this section, we briefly review related existing work related to our method from two aspects: point cloud compression and deep learning for point cloud segmentation.

A. Point Cloud Compression

As for point cloud compression problem in 3D LiDAR, most researchers focus on data compression rather than reduction. Height map-based methods have been studied in point cloud compression. [15] was the first to propose using height maps to compress point cloud data, and following research like [1]–[3] were then developed in response. However, converting a 3D point cloud into a 2D image will inevitably result in information loss. [4]–[6] used the raw packet data from LiDAR and rearranged them to a 2D matrix, and then using existing image methods to process the data. [16] proposed a recurrent neural network for point cloud encoding to accomplish compression. [17] designed a learning-based auto-encoder framework by hybrid representation of adaptive octree and bitstream specifications. These previous researches are not localization-oriented and can be used to complementing the task in this paper.

Effective reduction of the map database by finding critical representations is a popular topic in the vision community. [9] proposed a scoring function using the observation count of landmarks for compression and found that landmarks with high scores served as informative parts in localization. [10], [11] proposed a prioritization scheme to match 2D images and 3D point cloud reconstructed by a collection of images. They show that selecting a reduced set of points with the high priority is a feasible compression strategy, as it covers more robot poses or keyframes using a small amount of points. [12] formulated this task as a mixed-integer quadratic programming problem. For 3D point clouds, [19], [20] leveraged salient regions extracted from 3D LiDAR data for scan matching or localization. [14] proposed an efficient linear programming method to generate compressed point cloud map for SLAM, and then trained a random forest model using the map generated above as supervision. However, it is [14] that can only tackle with point cloud maps. In other words, this method needs global information before compression. In this paper, our PocoNet takes only one frame point clouds from LiDAR to accomplish SLAM-oriented compression or reduction, only relying on single-frame geometric information rather than global.

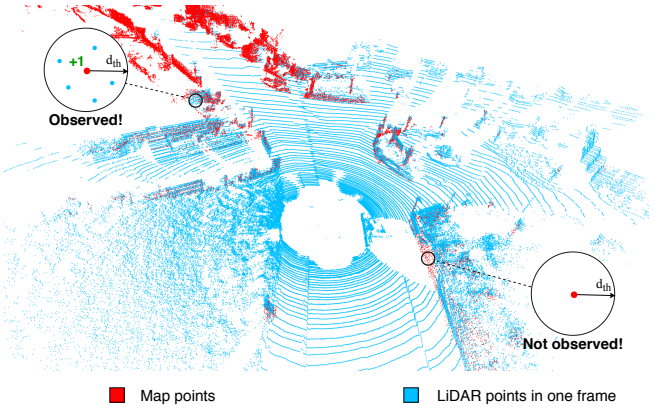


Fig. 2: Illustration of calculating observation counts for each map point at one pose. If there are one or more LiDAR points within the threshold radius of a map point d_{th}, the observation count of this map point will be increased by one.

B. Deep Learning for Point Cloud Segmentation

PointNet [21] pioneered the direct processing of raw point clouds. Inspired by the method above, many recent works introduced learning-based methods to capture geometric features of points. PointNet++ [22] suggested a hierarchical application of PointNet to explore local structures. PVCNN [23] improved the efficiency of PointNet-based methods using voxel-based convolution with a contiguous memory pattern. Such approaches, however, have been limited to address with small-scale point clouds like object parts and indoor scenes. For large-scale point clouds like outdoor scenes, it is difficult for methods mentioned above to satisfy a real-time latency constraint. SqueezeSeg [24]–[26] and RangeNet++ [27] utilized the spherical projection mechanism to project 3D point clouds to 2D and applied different convolution operations on projected 2D images. [28] followed bird-view projection and used polar coordinate system for encoding system. For processing raw point clouds, it is noteworthy of mentioning that the most recent RandLA-Net [18] significantly improved the speed of point cloud processing in the novel use of random sampling. As a consequence, our PocoNet takes RandLA-Net [18] as the baseline to learn geometric features.

III. Methodology

A. Problem Statement

We first formulate SLAM-oriented 3D LiDAR point cloud compression as the task of point cloud segmentation. The given point cloud is denoted as $\mathbf{P}$ which is a point set in a

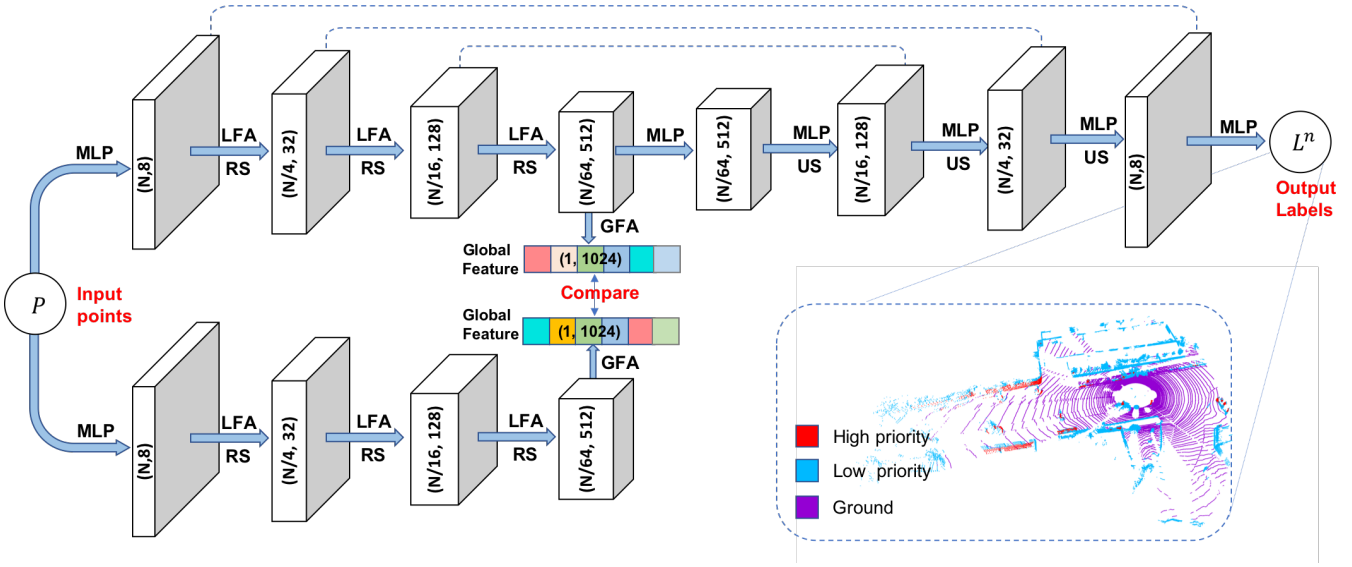


Fig. 3: PocoNet Architecture. PocoNet takes all points of one frame data from LiDAR as input and uses Local Feature Aggregation(LFA) Modules and Random Sampling (RS) in line with RandLA-Net [18] to encode point features. Our Global Feature Attention Module (GFA) then outputs a global feature codeword with self-attentions. Siamese encoder is utilized to blunt the impact of random sampling for global feature encoding. Finally, PocoNet outputs point-wise labels (high priority, low priority, and ground).

data frame from LiDAR containing n points $p_1, p_2, \ldots, p_n \in \mathbb{R}^d$. For 3D LiDAR sensors, the feature vector of each point p_i is its coordinate in 3D space plus its laser intensity value, $(x_i, y_i, z_i, intensity_i)$. The set of annotated labels is denoted as $\mathbf{L}$, including three categories: high priority, low priority, and ground, respectively. Points marked as reserved with high priority that they act as a primary part for localization.

The segmentation of a point cloud for compression is a function Ψ which assigns labels mentioned above to each point in the point cloud. i.e.:

$$\Psi : \mathbf{P} \longmapsto \mathbf{L}^n \tag{1}$$

The objective of this segmentation algorithm is finding optimal function Ψ that selects a subset of critical representative points. After this procedure, every point in a point cloud $\mathbf{P}$ will be assigned a label mentioned above. The point cloud compression is carried out by retaining most points marked as high priority and some ground points, while we remove others, to generate compressed point clouds $\mathbf{P}_c$, as follows:

$$\mathbf{P} \xrightarrow{\mathbf{L}^n} \mathbf{P}_c \tag{2}$$

B. Point Cloud Annotation

Here we introduce a method based on observation counts to annotate labels for each point from a LiDAR scan.

The evaluation of localization performance can be carried out by calculating the matching accuracy between the map and point clouds. In order to measure the effect on localization performance of each point from one frame LiDAR scan, it is essential to score the point cloud map by observation counts. Thus, we construct the map using a sequence of vehicle poses $\mathbf{T} = \{t_i\} \in SE(3)$ and their corresponding laser scans $\mathbf{S} = \{s_i\}$, where s_i represents the point cloud of a raw scan from LiDAR. The full map is denoted by $\mathbf{M_o} = \{m_i\}$, where m_i stands for the i_{th} point in the map.

Following the scoring method from [14], the definition of whether a map point m_i is observed or not derives from point cloud matching between $\mathbf{M_o}$ and $\mathbf{S}$. Here we utilize iterative closest point (ICP) as our matching algorithm, which is one of the most commonly used point cloud registration approaches in SLAM. Here we calculate the 3D Euclidean distance d_{m_i,s_j} between m_i and the closest point to m_i in s_j. Based on the matched distance d_{m_i,s_j} after the pose is estimated, we set the map point as observed with the following criteria:

$$\text{Observed} = \begin{cases} 0 & d_{m_i,s_j} \geq d_{th} \\ 1 & d_{m_i,s_j} < d_{th} \end{cases} \tag{3}$$

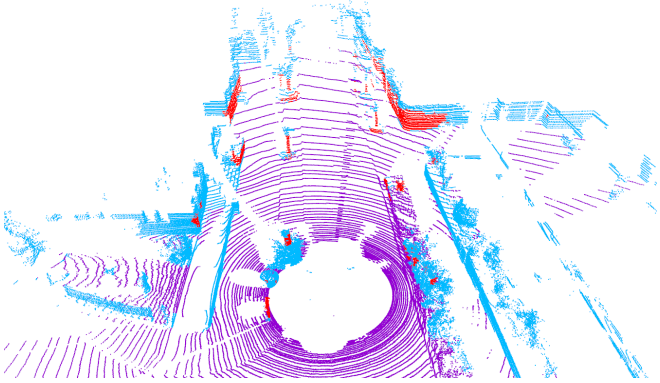
where d_{th} is a threshold value to decide if m_i at one pose is observed or not. To reduce the sensitivity to high-density points, we only count once for m_i at one pose, although it may be observed by different s_i. Given a sequence of poses $\mathbf{T}$ and raw scans $\mathbf{S}$ from LiDAR, we can score each map point m_i by aligning the scans to the map and cumulating the observation counts. As shown in Fig. 2, the scoring process should fulfil the following strategies:

- A map point cannot be observed multiple times repeatedly at one pose.
- The observation counts can be accumulated in different poses.

Finally, all map points $\{m_i\}$ have their observation score $\mathbf{C}^m = \{c_i^m\}$. The larger count is, the more times a landmark is observed, the more useful for localization. Further, we conduct nearest-neighbour searches to assign these observation $\mathbf{C}^m$ from the map $\mathbf{M_o}$ to $\mathbf{S}$. Then, we generate the observation score of raw scans $\mathbf{S}$, which is denoted as $\mathbf{C} = \{c_i\}$.

It is worth emphasizing that points on the ground should be individually marked. A possible explanation for this is that ground points, compared to the number of them, only provide

Camera View



$\mathbf{P}$ labelled from PocoNet

Fig. 4: Visualization of points with high priority (colored in red) on a original LiDAR scan. Intuitively, PocoNet mostly selects points like tree trunks, walls and some other landmarks to reserve, however, without semantic information.

plane information on localization in SLAM application. In addition, due mainly to high densities of ground points, the observation counts of them usually have large values, which harms our label annotation. The ground segmentation can utilize traditional methods like RANSAC [29].

Besides points marked as ground, the remaining points are required to be labelled based on whether their observation counts are greater than the threshold value c_{th}. Considering that observation counts are directly related to the trajectory, zero or low speed situation in data collection could result in uneven distribution of annotated labels in $\mathbf{S}$. In order to address this problem, we present a novel strategy to set the threshold value dynamically, illustrated in Alg. 1. This strategy ensures that the number of reserved points remains consistent between different LiDAR scans and avoids the influence of the trajectory for label annotation. After these procedures above, points with annotated labels $\mathbf{L}^n$ are provided to the proposed network as the training data.

C. PocoNet Architecture

1) Overview: Now we introduce the PocoNet in detail, where the network architecture is illustrated in Fig. 3. Given a 3D point cloud from LiDAR, PocoNet network aims to assign point-wise priority labels (high priority, low priority, and ground) to accomplish SLAM-oriented point cloud compression, as shown in Fig. 4. To capture geometric feature from point clouds in real-time, PocoNet's backbone architecture is based on RandLA-Net [18]. It, due to progressively capturing features of growing scales in a hierarchy approach, is capable of accomplishing real-time segmentation for large-scale points, which matches with the demands of our task.

Note that RandLA-Net [18] has not yet been explicitly leveraging global geometric context. However, capturing global geometry information acts as a primary factor since our task is to select a subset from all. To this end, we propose the Global Feature Attention (GFA) module to extract global geometric contextual encoding. Besides, two legs of encoder sharing weights (i.e. siamese encoder) is employed to blunt the impact of random sampling for encoding global features.

2) Global Feature Attention Module: Traditional methods like [21] leverage Multi-Layer-Perceptron (MLP) layers to abstract each feature into higher dimension individually and a concise max-pooling operation to aggregate them into a global feature codeword. However, these two simple operations can hardly capture the correlation in the feature space. Inspired by [30] [31], the attention mechanism is suitable for learning the correlation between features. Here we propose the Global Feature Attention module following the self-attention mechanism to highlight the different importance of each point for better perceiving global geometric information.

Algorithm 1 Dynamic Threshold Setting Strategy

Input:
The matrix of observation counts of points, $\mathbf{C} \in \mathbb{R}^{N\times 1}$;
The threshold value of reserved percentage, r_{th};
Output:
The threshold value of the observation count, c_{th};
The minimum number of reserved points, $N_{th} \Leftarrow N * r_{th}$;
The maximum observation count in $\mathbf{C}$, c_{max};
$c_{th} \Leftarrow c_{max}$;
$N_{greater} \Leftarrow N$;
while $N_{greater} < N_{th}$ **do**
  Calculate the number of points whose observation count greater than c_{th}, $N_{greater}$;
  $c_{th} \Leftarrow c_{th} - 1$;
end while
return c_{th};

The feature map x is first transformed into two feature spaces f and g to calculate the attention map, where $f(x) = W_f x$, $g(x) = W_g x$

$$\beta_{i,j} = \frac{\exp\left(s_{ij}\right)}{\sum_{i=1}^{D_1} \exp\left(s_{ij}\right)}, \text{ where } s_{ij} = f\left(x_i\right)^T g\left(x_j\right) \quad (4)$$

and $\beta_{i,j}$ evaluates the correlation which the model pays to the i^{th} point when considering the j^{th} feature vector. Then the attention result is $r = (r_1, r_2, \cdots, r_j, \cdots, r_{D_1}) \in \mathbb{R}^{D_1 \times D_2}$, where

$$r_j = \sum_{i=1}^{D_1} \beta_{j,i} h\left(x_i\right), \text{ where } h\left(x_i\right) = W_h x_i \quad (5)$$

In the above formulation, W_f, W_g, W_h are learned weight matrices, which are implemented as 1×1 convolutions.

Furthermore, we further concatenate the result above with the input feature matrix, denoted by $o_i = x_i \oplus r_i$, where $\oplus$ is the concatenation operation. This allows the network to rely on the cues among the feature vectors. Finally, a MLP block and a max-pooling operation are employed to integrate the multiple features with correlation information. Via hierarchically abstracting features from local to global, GFA module generates a 1024-dimension codeword to represent the global geometric feature.

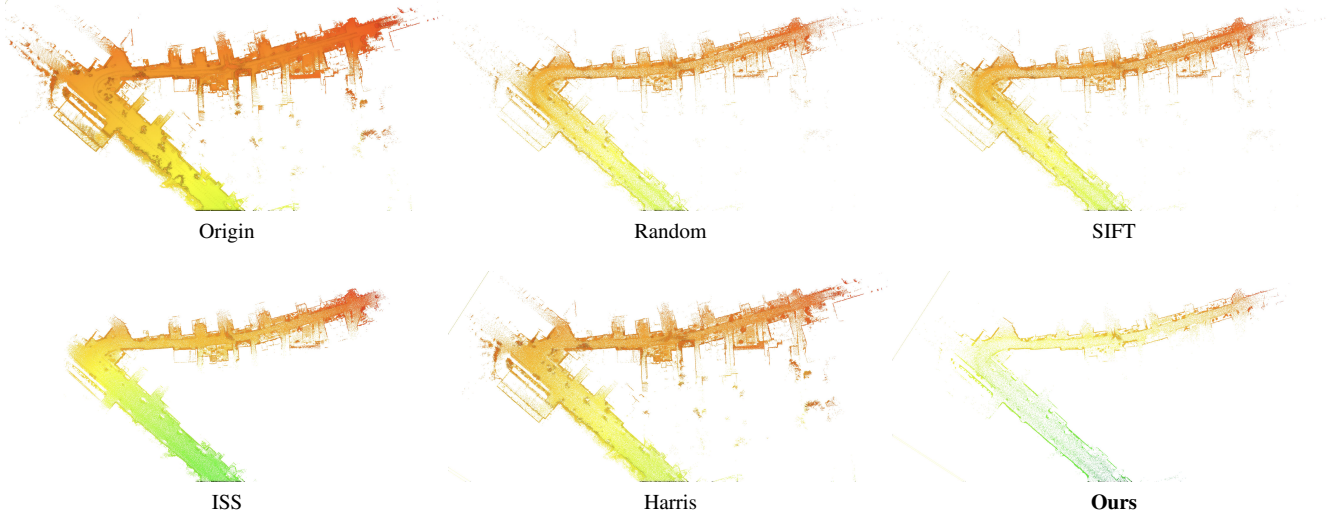


Fig. 5: Visualization of generated maps with using compressed point clouds. (Best viewed with zoom-in.)

3) Loss Funtion: The loss function consists of the global encoding loss $\mathcal{L}_{\text{global}}$ and compression label loss $\mathcal{L}_{\text{label}}$, defined as

$$\begin{aligned} \mathcal{L}_{\text{label}} &= -\mathbf{L}_n \log \hat{\mathbf{L}}_n - (1-\mathbf{L}_n) \log\left(1-\hat{\mathbf{L}}_n\right) \\ \mathcal{L}_{\text{global}} &= \|V_1 - V_2\|_2 \end{aligned} \tag{6}$$

$\mathbf{L}_n$ stands for labels annotated in III-B to supervise the predicted labels $\hat{\mathbf{L}}_n$. Global feature codewords generated from two branches of shared-weighted encoder is denoted as V_1 and V_2. For the purpose of blunting the impact of random sampling, here we use element-wise L_2 loss for global feature encoding to penalize the distance between V_1 and V_2. Besides, the cross-entropy loss is applied between $\mathbf{L}_n$ and $\hat{\mathbf{L}}_n$. The overall loss function is defined as

$$\mathcal{L} = \mathcal{L}_{\text{label}} + \lambda \cdot \mathcal{L}_{\text{global}} \tag{7}$$

where λ is the weighting factor.

IV. Experiments

We conduct experiments by integrating our compression module into an existing SLAM system [14] evaluate compression rates and localization performances. The experimental evaluation is designed to support the critical claims that our approach can: (i) predict point-wise priority with only single-frame data in real-time, (ii) represent the whole trajectory with fewer points, (iii) maintain localization accuracy after compression, (iv) achieve generalization to new environments.

A. Dataset and Implementation Details

In this section, we evaluate the point cloud compression on two popular large-scale autonomous driving datasets: KITTI [32] and Ford Campus [33]. In our experiments, the sequences 00 $\sim$ 08 in KITTI is annotated to serve as supervision for training, and 09 $\sim$ 10 for testing. Besides, sequence 00 in Ford Campus is used for experiments of generalization. The raw 3D points only have 3D coordinates and intensity values without color information. Considering application scenarios, we set the threshold value of reserved percentage r_{th} as $\frac{1}{12}$, which means that the number of reserved points generally remains around 10000 for one frame data. The threshold of matched distance for counting observation counts is set as $d_{th} = 0.1m$. Here we use the Adam optimizer with default parameters with an initial learning rate is set as 10^{-3} and decreases by 5% after each epoch. The number of nearest points k is established as 12. To train our PocoNet in parallel, we sample a fixed number of points (4096 * 30) as the input. If the number of point clouds from one frame is less than that fixed number, we will randomly duplicate some points to keep the quantity consistent. All experiments are conducted on an NVIDIA GTX1080Ti GPU.

B. Evaluation Metric

In order to evaluate the performance of the models in terms of preserving localization accuracy after compression, we conduct localization experiment (i.e. pose tracking) and report the compressed sizes and localization errors to demonstrate the effectivity of the proposed network. The pose of the first frame is given as the fixed start position. Point-to-plane ICP is applied for each compressed laser scan, which uses the previous result as the initial value to achieve pose tracking. We generate maps from compressed point clouds with ground truth, and then employ the octree grid filter following the actual operations. Localization tests are conducted between maps generated above and compressed scans. As for the registration process, we use the same set of parameters for fair testing. To our best knowledge, no previous work has integrated LiDAR localization with single-frame point cloud compression. It is noted that 3D keypoint detection conforms with our task, since both tasks select a subset of points to represent the whole point cloud for efficient matching. Hence, we generate compressed point clouds using these keypoint detection methods: Harris-3D [34], SIFT [35], Intrinsic Shape Signatures (ISS) [36]. Besides, random sampling and original point clouds (denoted by Origin) are also evaluated. For the fair comparison, the parameters of different methods are dynamically set to keep

	Seq 09					*Seq 10*				
	$Trans\ mATE(m)$	$Rot\ mAPE$	$mAPE$	N	r_c	$Trans\ mATE(m)$	$Rot\ mAPE$	$mAPE$	N	r_c
Origin	0.02065	0.00702	0.02240	17,066,008	43.90%	0.02442	0.00564	0.02563	7,385,291	28.16%
Random	0.03935	0.01156	0.04191	1,742,508	4.48%	0.04474	0.01132	0.04700	1,097,402	4.18%
Harris [34]	0.06147	0.00923	0.06246	2,401,497	6.18%	0.04738	**0.00608**	0.04799	1,159,490	4.42%
SIFT [35]	0.03382	0.01113	0.03686	1,458,835	3.75%	0.04275	0.00733	0.04384	1,466,450	5.59%
ISS [36]	0.04583	0.01216	0.04895	1,928,035	4.96%	0.06068	0.01028	0.06232	904,020	3.44%
Ours	**0.02488**	**0.00888**	**0.02710**	**874,083**	**2.25%**	**0.03681**	0.00998	**0.03918**	**377,325**	**1.44%**

TABLE I: Quantitative results of localization performances on KITTI [32].

the number of points in one frame after compression in the same quantity range ($8000 \sim 10000$).

C. Localization Performance

The results of localization tests are shown in Tab. I and Fig. 6. We compare the 6D registration results with the ground true poses to report Translation mean Absolute Trajectory Error ($Trans\ mATE$), Rotation mean Absolute Trajectory Error ($Rot\ mAPE$) and mean Absolute Pose Error ($mAPE$) which considers both translation and orientation error. The number of points of maps (N) and compression ratios (r_c) from the full map without octree downsampling are also demonstrated.

Compared to others, our PocoNet achieves the highest compression rate, i.e., the minimum number of points of maps (visualized in Fig. 5). Besides, PocoNet generally performs better for localization than other comparative methods. We attribute this result to the following aspects. To begin with, our PocoNet retains the points that can be reused many times in global localization. For this reason, our method, compared with other non-SLAM-oriented approaches, can achieve a higher compression rate. Furthermore, as for the localization performance, landmarks with more observation counts has better consistency between scan frames, which are favourable to maintaining localization accuracy.

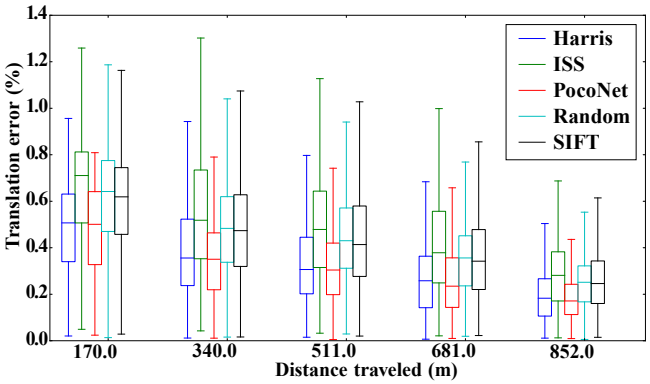


Fig. 6: Boxplot of the relative trajectory error statistics in 09. The middle box spans the first and third quartiles, while the whiskers are the upper and lower limits. Plot best seen in color.

D. Computational Efficiency

Hand-craft detectors are deployed with single thread C++ codes using PCL [37]. Our proposed method is deployed with Pytorch [38]. Here the computational efficiency is evaluated by recording the average time taken to extract compressed points in the same quantity range ($8000 \sim 10000$). As shown in Tab. II, our PocoNet is an order of magnitude faster than other methods except random sampling.

E. Generaization Ability

To highlight the generalization capabilities of our approach, we evaluate our model on the Ford campus dataset while using only KITTI for training. The Ford campus dataset [33] is recorded on the Ford research campus and downtown Dearborn in Michigan using a different version of Velodyne HDL-64E. Here we use a sequence in 01 for testing our model. Note that we never trained our approach on the Ford campus dataset or even US roads. Tab. III demonstrates that our PocoNet also preserves localization accuracy with fewer points of the map.

Method	Random	ISS	SIFT	Harris	Ours
Average Time (s)	0.0005	1.8249	7.6425	1.0927	**0.0460**

TABLE II: Average time to extract the compressed point clouds in the same quantity range from KITTI, respectively.

	$Trans\ mATE(m)$	$Rotation\ mAPE$	$mAPE$	N	r_c
Origin	0.01962	0.00158	0.01976	6,552,607	23.68%
Random	0.03353	0.00392	0.03401	1,452,985	5.25%
Harris [34]	0.04007	**0.00283**	0.04027	1,490,501	5.39%
SIFT [35]	0.02606	0.00495	0.02684	1,298,121	4.69%
ISS [36]	0.04245	0.00389	0.04274	1,266,933	4.58%
Ours	**0.02525**	0.00340	**0.02566**	**1,168,869**	**4.22%**

TABLE III: Quantitative results of localization performances on Ford Campus [33].

V. Conclusions

In this paper, we presented a novel approach for SLAM-oriented point cloud compression. The key insight is that points with high priority have similar geometric features in SLAM scenarios. We regard this task as segmentation on point clouds to capture complex geometric information. We calculate the observation counts by matching between maps and point clouds and divide these counts into priority levels. Trained by labels annotated with such observation counts, the proposed network could evaluate the priority level of single-frame point clouds. Exhaustive evaluations on two different datasets show that our PocoNet achieves the highest compression rate with better localization accuracy, which demonstrates the feasibility and generalization ability of our approach in real robotics applications. Encouraged by our results, we are considering several avenues to continue exploring the relationship between priority levels and geometric features of point clouds in SLAM applications.

REFERENCES

[1] E. Hubo, T. Mertens, T. Haber, and P. Bekaert, "Self-similarity based compression of point set surfaces with application to ray tracing," *Computers & Graphics*, vol. 32, no. 2, pp. 221–234, 2008. 1, 2

[2] T. Ochotta and D. Saupe, *Compression of point-based 3D models by shape-adaptive wavelet coding of multi-height fields*, 2004. 1, 2

[3] ——, "Image-based surface compression," in *Computer graphics forum*, vol. 27, no. 6. Wiley Online Library, 2008, pp. 1647–1663. 1, 2

[4] H. Yin and C. Berger, "Mastering data complexity for autonomous driving with adaptive point clouds for urban environments," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1364–1371. 1, 2

[5] C. Tu, E. Takeuchi, C. Miyajima, and K. Takeda, "Compressing continuous point cloud data using image compression methods," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1712–1719. 1, 2

[6] ——, "Continuous point cloud data compression using slam based prediction," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2017, pp. 1744–1751. 1, 2

[7] E. Hubo, T. Mertens, T. Haber, and P. Bekaert, "The quantized kd-tree: Efficient ray tracing of compressed point clouds," in *2006 IEEE Symposium on Interactive Ray Tracing*. IEEE, 2006, pp. 105–113. 1

[8] R. Schnabel and R. Klein, "Octree-based point-cloud compression." *Spbg*, vol. 6, pp. 111–120, 2006. 1

[9] M. Dymczyk, S. Lynen, T. Cieslewski, M. Bosse, R. Siegwart, and P. Furgale, "The gist of maps-summarizing experience for lifelong localization," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2767–2773. 1, 2

[10] Y. Li, N. Snavely, and D. P. Huttenlocher, "Location recognition using prioritized feature matching," in *European conference on computer vision*. Springer, 2010, pp. 791–804. 1, 2

[11] Y. Li, N. Snavely, D. Huttenlocher, and P. Fua, "Worldwide pose estimation using 3d point clouds," in *European conference on computer vision*. Springer, 2012, pp. 15–29. 1, 2

[12] H. Soo Park, Y. Wang, E. Nurvitadhi, J. C. Hoe, Y. Sheikh, and M. Chen, "3d point cloud reduction using mixed-integer quadratic programming," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, 2013, pp. 229–236. 1, 2

[13] S. Lynen, T. Sattler, M. Bosse, J. A. Hesch, M. Pollefeys, and R. Siegwart, "Get out of my lab: Large-scale, real-time visual-inertial localization." in *Robotics: Science and Systems*, vol. 1, 2015. 1

[14] H. Yin, Y. Wang, L. Tang, X. Ding, S. Huang, and R. Xiong, "3d lidar map compression for efficient localization on resource constrained vehicles," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–16, 2020. 1, 2, 3, 5

[15] M. Pauly and M. Gross, "Spectral processing of point-sampled geometry," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 379–386. 2

[16] C. Tu, E. Takeuchi, A. Carballo, and K. Takeda, "Point cloud compression for 3d lidar sensor using recurrent neural network with residual blocks," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 3274–3280. 2

[17] X. Wen, X. Wang, J. Hou, L. Ma, Y. Zhou, and J. Jiang, "Lossy geometry compression of 3d point cloud data via an adaptive octree-guided network," in *2020 IEEE International Conference on Multimedia and Expo (ICME)*, 2020, pp. 1–6. 2

[18] 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," *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2020. 2, 3, 4

[19] D. Cole, A. Harrison, and P. M. Newman, "Using naturally salient regions for slam with 3d laser data," in *International Conference on Robotics and Automation, SLAM Workshop*, 2005. 2

[20] Y.-J. Lee, J.-B. Song, and J.-H. Choi, "Performance improvement of iterative closest point-based outdoor slam by rotation invariant descriptors of salient regions," *Journal of intelligent & robotic systems*, vol. 71, no. 3-4, pp. 349–360, 2013. 2

[21] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," pp. 652–660, 2017. 2, 4

[22] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108. 2

[23] Z. Liu, H. Tang, Y. Lin, and S. Han, "Point-voxel cnn for efficient 3d deep learning," in *Advances in Neural Information Processing Systems*, 2019, pp. 965–975. 2

[24] 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 *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2018, pp. 1887–1893. 2

[25] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "Squeezesegv2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a lidar point cloud," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 4376–4382. 2

[26] C. Xu, B. Wu, Z. Wang, W. Zhan, P. Vajda, K. Keutzer, and M. Tomizuka, "Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation," *arXiv preprint arXiv:2004.01803*, 2020. 2

[27] 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. 2

[28] Y. Zhang, Z. Zhou, P. David, X. Yue, Z. Xi, B. Gong, and H. 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*, 2020, pp. 9601–9610. 2

[29] R. Schnabel, R. Wahl, and R. Klein, "Efficient ransac for point-cloud shape detection," in *Computer graphics forum*, vol. 26, no. 2. Wiley Online Library, 2007, pp. 214–226. 4

[30] H. Zhang, I. Goodfellow, D. Metaxas, and A. Odena, "Self-attention generative adversarial networks," *arXiv preprint arXiv:1805.08318*, 2018. 4

[31] X. Liu, Z. Han, X. Wen, Y.-S. Liu, and M. Zwicker, "L2g auto-encoder: Understanding point clouds by local-to-global reconstruction with hierarchical self-attention," 2019. 4

[32] 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. 5, 6

[33] G. Pandey, J. R. McBride, and R. M. Eustice, "Ford campus vision and lidar data set," *The International Journal of Robotics Research*, vol. 30, no. 13, pp. 1543–1552, 2011. 5, 6

[34] C. G. Harris, M. Stephens *et al.*, "A combined corner and edge detector." in *Alvey vision conference*, vol. 15, no. 50. Citeseer, 1988, pp. 10–5244. 5, 6

[35] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, no. 2, pp. 91–110, 2004. 5, 6

[36] Y. Zhong, "Intrinsic shape signatures: A shape descriptor for 3d object recognition," in *2009 IEEE 12th International Conference on Computer Vision Workshops, ICCV Workshops*. IEEE, 2009, pp. 689–696. 5, 6

[37] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *2011 IEEE international conference on robotics and automation*. IEEE, 2011, pp. 1–4. 6

[38] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems*, 2019, pp. 8024–8035. 6