A Task-Driven Scene-Aware LiDAR Point Cloud Coding Framework for Autonomous Vehicles

Xuebin Sun, Miaohui Wang, *Member, IEEE*, Jingxin Du, Yuxiang Sun, Shing Shin Cheng, *Member, IEEE*, and Wuyuan Xie, *Member, IEEE*

***Abstract*—LiDAR sensors are almost indispensable for autonomous robots to perceive the surrounding environment. However, the transmission of large-scale LiDAR point clouds is highly bandwidth-intensive, which can easily lead to transmission problems, especially for unstable communication networks. Meanwhile, existing LiDAR data compression is mainly based on rate-distortion optimization, which ignores the semantic information of ordered point clouds and the task requirements of autonomous robots. To address these challenges, this article presents a task-driven Scene-Aware LiDAR Point Clouds Coding (SA-LPCC) framework for autonomous vehicles. Specifically, a semantic segmentation model is developed based on multidimension information, in which both 2-D texture and 3-D topology information are fully utilized to segment movable objects. Furthermore, a prediction-based deep network is explored to remove the spatial–temporal redundancy. The experimental results on the benchmark semantic KITTI dataset validate that our SA-LPCC achieves state-of-the-art performance in terms of the reconstruction quality and storage space for downstream tasks. We believe that SA-LPCC jointly considers the scene-aware characteristics of movable objects and removes the spatial–temporal redundancy from an end-to-end learning mechanism, which will boost the related applications from algorithm optimization to industrial products.**

***Index Terms*—Autonomous vehicles, LiDAR point clouds, semantic segmentation.**

I. Introduction

LIDAR sensors can directly describe the real world through point clouds with precise scale measurements and geometric features, which have extensively promoted the sensing module of autonomous driving and robotics [1]. It has become an indispensable module for perceiving the external environment [2]. The remote transmission of large-scale LiDAR point clouds is time-consuming, especially when the network link is unstable. Therefore, it is a crucial task to reduce the storage space and transmission bandwidth of LiDAR data for downstream 3-D applications [3], [4].

LiDAR data [5] consist of a set of points in the 3-D space, represented by their coordinates. To provide a more realistic representation, each data point can have multiple attributes, including color, normal vector, and reflectivity. Previously, LiDAR data are usually represented by a tree structure (e.g., Octree and *k*-d tree). For instance, the Octree structure is to recursively divide a point into eight Octants, in which each internal node has exactly eight children. In addition, quantization is usually employed to reduce the storage size. However, this data representation rarely takes advantage of the fact that LiDAR data are well-geometrically structured. There is still a lot of redundant information hidden in this representation, such as repeated local structures, planes, or objects with strong shapes (e.g., cars and humans).

Recently, deep-learned data compression has become a hot research topic, which removes redundancy through a compact feature representation. Some early studies [6], [7] have been developed for LiDAR data encoding. These methods usually involve the following steps:

1) encoding data into hidden feature representations via convolutional neural networks (CNN);
2) quantizing the hidden features;
3) learning an entropy model, which further compresses the bitstream via the entropy coding.

At present, most deep-learning-based compression algorithms have been developed for dense point clouds, primarily focusing on eliminating spatial redundancy. However, 3-D LiDAR data compression is still challenging due to its sparsity, irregularity, disorder, and rotation invariance. Therefore, further exploration is essential for the application of autonomous driving.

Manuscript received 20 June 2022; revised 25 September 2022; accepted 6 November 2022. Date of publication 10 November 2022; date of current version 13 July 2023. This work was supported in part by the National Natural Science Foundation of China under Grant 61701310 and Grant 62106152, in part by the Natural Science Foundation of Guangdong Province under Grant 2022A1515011245 and Grant 2019A1515010961, and in part by the Natural Science Foundation of Shenzhen City under Grant 20220809160139001 and Grant 20200805200145001. Paper no. TII-22-2629. *(Corresponding author: Miaohui Wang.)*

Xuebin Sun and Miaohui Wang are with the Guangdong Key Laboratory of Intelligent Information Processing, Shenzhen University, Shenzhen 518060, China, and also with the Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen 518172, China (e-mail: sunxuebin@szu.edu.cn; wang.miaohui@gmail.com).

Jingxin Du and Shing Shin Cheng are with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong Shatin, Hong Kong, SAR, China (e-mail: jingxindu@cuhk.edu.hk; sscheng@cuhk.edu.hk).

Yuxiang Sun is with the Department of Mechanical Engineering, The Hong Kong Polytechnic University, Hong Kong, SAR, China (e-mail: yxsun@polyu.edu.hk).

Wuyuan Xie is with the College of Computer Science and Software Engineering, Shenzhen University, Shenzhen 518060, China (e-mail: wuyuan.xie@gmail.com).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TII.2022.3221222.

Digital Object Identifier 10.1109/TII.2022.3221222

In addition, to realize L4 unmanned driving [8] in the actual environment, it is necessary to build a LiDAR point cloud map covering the street in advance as the prior information of position and navigation. However, in the simultaneous location and mapping, it is well known that so-called damage points are the relevant data associated with movable objects [3], [9] (e.g., pedestrians and vehicles). Since these damaged points provide "noise" information in localization [10], these data points should be removed in the coding stage to improve the accuracy of mapping and positioning. Unfortunately, this problem has not been well investigated in existing LiDAR data compression algorithms.

To perceive the surrounding environment under different lighting conditions (daytime or nighttime) or different weather conditions (foggy, snowy, windy, or rainy), LiDAR sensors are almost indispensable for autonomous robots. However, the transmission and storage of massive point clouds are highly time-consuming and resource-consuming. Meanwhile, existing LiDAR data processing mainly treats compression as a rate-distortion optimization problem, and seldom considers the semantic features of LiDAR point clouds. To tackle the above issues, in this article, we introduce a task-driven Scene-Aware LiDAR Point Clouds Coding (SA-LPCC) framework for autonomous vehicles. The main contributions are as follows:

1) We develop a multimodal fusion-based LiDAR point cloud semantic segmentation network to identify moving objects (e.g., pedestrians and vehicles), which paves the way for further scene-aware LiDAR coding.
2) We propose a new hybrid LiDAR data encoding framework. Specifically, we explore a plane-fitting-based method to predict the ground, and investigate a deep network model to reconstruct the spatial structure of point clouds to largely remove spatial–temporal redundancy.
3) To our knowledge, this is a pioneer task-driven scene-aware method for LiDAR point clouds. Experimental results on the benchmark dataset demonstrate the superiority of our SA-LPCC in maintaining the high compression quality as well as the low storage space.

The rest of this article is organized as follows. In Section II, we briefly review some relevant studies on LiDAR point clouds segmentation and compression. In Section III, we provide an overview of our SA-LPCC method. In Sections IV and V, we present the task-driven LiDAR point cloud semantic segmentation and our hybrid encoding network, respectively. We present the experimental results in Section VI. Finally, Section VII concludes this article.

II. Related Work

Considering the topic of this article, we briefly review the relevant LiDAR point cloud semantic segmentation and coding methods in this section.

A. Semantic Segmentation on Point Clouds

1) Projection- and Voxel-Based Methods: To leverage the success of CNN, 3-D point clouds have been transformed into 2-D images to address the task of segmentation [11], [12], [13], [14], [15], [16]. For example, Wu et al. [11] utilized *SqueezeNet* to realize point cloud segmentation and employed a conditional random field to refine the related label map. By improving the model structure, training loss, and input channel, they further developed a new model *SqueezeSegV3* [13], which was more robust to dropout noises in LiDAR point clouds. *RangeNet++* [14] employed a range image as an intermediate representation to train a deep-learned segmentation model for point cloud data. *PolarNet* [16] projected point clouds into a bird-eye view image and converted the quantized points into a polar coordinate for semantic segmentation. *FPS-Net* [15] adopted an encoder–decoder structure to explore the uniqueness and discrepancy between 2-D projected representations.

Currently, most existing methods have been developed for dense 3-D point clouds, but they rarely consider the distinct characteristics of LiDAR data due to its sparsity, irregularity, and scan manner. This is one of the fundamental motivations to explore a special semantic segmentation scheme for LiDAR point clouds by considering both 3-D features and 2-D multimodal features.

2) Point-Based Schemes: Some point-based segmentation networks have been explored to directly segment original 3-D point clouds [17], [18], [19]. One pioneering effort *PointNet* [17] extracted point-by-point features from unordered point cloud data via a shared multilayer perception (MLP). Considering that *PointNet* was inadequate to extract local features of point clouds, Zhao et al. [18] introduced an attention mechanism to capture more local information. Furthermore, *RandLA-Net* [19] has been developed to utilize a local feature aggregation module to directly forecast per-point semantics for large-scale point clouds by random sampling.

Although existing point-based segmentation methods have achieved promising results, how to effectively segment large-scale LiDAR point clouds still requires more efforts.

B. LiDAR Point Clouds Coding

1) Tree-Based Coding Methods: Tree structure is one of the main data representations for existing point cloud compression algorithms. Many methods store point cloud data in an *Octree* and perform entropy encoding [20], [21], [22]. Queiroz et al. [20] proposed a context-driven algorithm based on the *Octree* structure. Instead of encoding a tree node with 1 b, the probability of each tree node was derived based on the distance between actual and reference voxels. Krivokuca et al. [21] introduced a volumetric-based approach for point clouds compression, which interpolated geometric and attribute values in encoding point data. Recently, Google's open-source project $Draco$ [22] used a k-d tree to perform the point compression.

The advantage of the *Octree* structure is its low encoding complexity. However, there is a lot of redundant information hidden in the repeated local structure, plane, or object with a strong shape prior, which still leaves room to explore advanced techniques to remove the relevant redundancy. It directly inspires this work to introduce semantic segmentation to improve the compression efficiency of LiDAR point data.

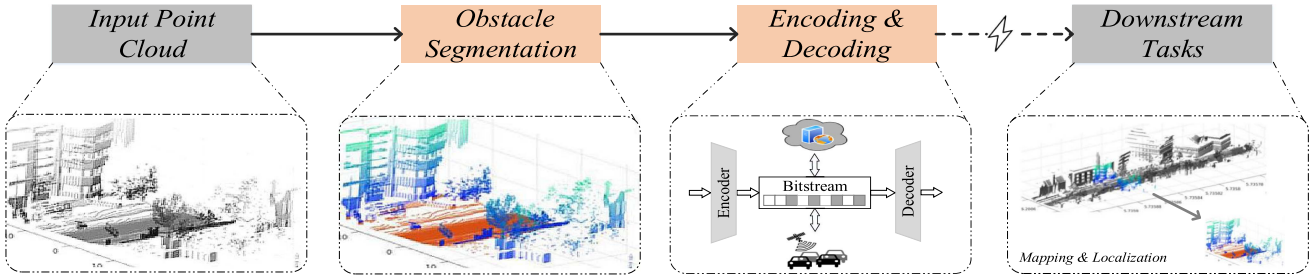


Fig. 1. Pipeline of the proposed task-driven SA-LPCC framework for autonomous vehicles. To eliminate the influence of movable objects, we develop a multimodal-based semantic segmentation scheme for LiDAR data. To reduce storage space and transmission bandwidth, we develop a plane-fitting-based method to remove spatial redundancy and investigate an unsupervised interframe network to remove temporal redundancy. Finally, we evaluate our SA-LPCC performance on some downstream tasks.

2) Hybrid Coding Schemes: Since the geometry of LiDAR scanning can be represented in a panoramic image, some studies have investigated the use of hybrid image coding techniques to reduce data redundancy in LiDAR data [7], [23], [24]. By converting 3-D point cloud data into a 2-D range image domain, Tu et al. [23] used the traditional image compression algorithm to encode the range values. Due to the regular geometric structure in point clouds, a clustering-based method has been utilized to remove the spatial redundancy of LiDAR data [7]. Feng et al. [24] developed a real-time spatiotemporal coding scheme for LiDAR point data. For intraframe coding, they utilized an iterative plane fitting to remove the spatial redundancy while for interframe coding, the point cloud registration technique was used to remove the temporal redundancy.

The main challenge for hybrid LiDAR data coding methods lies in rarely considering the 3-D characteristics of point clouds. For example, pedestrian and vehicle points are regarded as noise data. If these movable points are removed before encoding, the compression efficiency will be greatly improved.

III. Overall of the Task-Driven Scene-Aware Codec Architecture

There are two popular solutions for autonomous driving [25], including "drive in sensor" and "drive in map." For "drive in sensor," autonomous vehicles make decisions and path planning based on LiDAR sensors, which initially know nothing about their surrounding environments. However, for "drive in map," the surrounding environment information will be built as *a priori* high-precision navigation map in advance. A vehicle is positioned and navigated based on real-time sensor data and prestored high-precision maps. In general, the "drive in map" scheme is adopted by most manufacturers, which provides more stable performance and easier-to-manage landings. Therefore, LiDAR data processing is a key module for autonomous driving.

Obstacle points, like pedestrians and vehicles, will degrade the processing of LiDAR point cloud maps. As they are removable, we can build a clean LiDAR map, which is utilized for position and navigation. Usually, since these points bring "noise" information during the mapping process, it is favorable to remove them in the encoding stage, saving the bit budget for transmission and improving the localization and path planning performance in downstream tasks.

Algorithm 1: Multidimensional Feature-Fusion-Based LiDAR Point Cloud Semantic Segmentation.

Input:
Input point cloud I.
Output:
Segmentation results P.
1: Convert point cloud I to 2-D matrices, namely M_{dep}, M_{coor}, and M_{int}.
2: Extract the 2-D features in (2)–(4), namely $\mathcal{T}_{\text{dep}}$, $\mathcal{T}_{\text{coor}}$, $\mathcal{T}_{\text{int}}$.
3: Extract the 3-D features in (7), namely $\mathcal{I}_{\text{Dec}}^{L}$.
4: Fuse 2-D and 3-D features in (9), namely $\mathcal{F}_{\text{fusion}}$.
5: Calculate the segmentation results I_{seg} by an improved SegNet.

To address these issues, we design a task-driven scene-aware LiDAR data processing framework for autonomous vehicles. The pipeline of our SA-LPCC is illustrated in Fig. 1, and mainly consists of three modules: 1) obstacle segmentation, 2) LiDAR point cloud codec, and 3) downstream tasks.

In SA-LPCC, a multidimensional feature-fusion-based LiDAR data segmentation network is explored, assigning the semantic label to each point. Pedestrian and vehicle points will be classified as obstacles, and a hybrid encoding framework is developed for LiDAR point clouds. The bitstream will be transmitted to an online cloud or stored on a server. In downstream tasks, a point cloud map will be reconstructed based on the decoded point clouds.

IV. Multimodal Fusion-Based LiDAR Data Semantic Segmentation

In this section, we present the details of the proposed multimodal fusion-based LiDAR point cloud segmentation network. First, a point cloud conversion is performed to obtain a 2-D matrix representation. Then, the 2-D matrix and 3-D point clouds are jointly fed into two channels to extract the related 2-D and 3-D feature representations, respectively. Finally, the multimodal features are fused to perform the semantic segmentation.

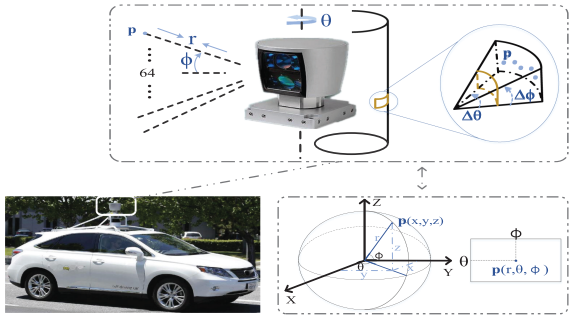


Fig. 2. Illustrations of a typical LiDAR sensor and the associated coordinate transformation.

A. Multimodal 2-D Feature Extractor

1) Conversion to 2-D Representation: For each scan of LiDAR point cloud $I_i \subset \mathbb{R}^{N_i \times 4}$, it has four attributes, including x, y, z, and intensity. Considering that LiDAR sensors acquire point data in an orderly manner, the resulted point clouds are converted from $\mathbb{R}^3$ to $\mathbb{R}^2$ in SA-LPCC. LiDAR data from the semantic KITTI dataset are utilized to verify our framework, which uses 64 channels, with a vertical field of view of 26.9° and a horizontal field of view of 360°. Let (x, y, z) be the coordinate in a frame of point clouds captured by *Velodyne* sensors, as shown in Fig. 2. Considering that point clouds are ordered, they can be converted to a 2-D matrix $M(\theta, \phi)$, and defined by

$$M\begin{pmatrix}\theta\\ \phi\end{pmatrix} = \begin{pmatrix}\frac{1}{2}(1-\arctan(y,x)\pi^{-1})\Theta\\ (1-(\arcsin(zr^{-1}+f_{up}))f^{-1})\Phi\end{pmatrix} \quad (1)$$

where (θ, ϕ) denotes the coordinate of a 2-D matrix, (Θ, Φ) denotes the height and width of M, and $r = ||\mathbf{p}_{x,y,z}||_2$ represents the distance between each point $\mathbf{p}_{x,y,z}$ and the coordinate origin. f_{up} and f_{down} denote the up bound and the low bound of vertical field-of-view f of the *Velodyne* sensor. In the experiments, the vertical angle between every two lines is fixed (e.g., 26.9°/64 beams = 0.42°) in each frame. In the horizontal direction, each point is also uniformly sampled, and the horizontal angle (e.g., 360°/2000 pixels = 0.18°) is treated as *a priori* information.

2) Multimodal 2-D Feature Fusion: For each point $\mathbf{p}_{x,y,z}$, we utilize its (x, y, z) coordinate, range r, and intensity to store them in a 2-D matrix form, creating $\Theta \times \Phi \times 5$ tensors: $\{M_{\text{dep}}, M_{\text{coor}}, M_{\text{int}}\}$. We develop a multikernel residual block (MKRB) to extract the 2-D feature representation from these tensors independently.

In MKRB, an input tensor is first fed to a parallel 1×1, 3×3, 5×5, and 7×7 convolution kernel to obtain four different multireceptive features. Unlike the conventional digital images, the projection matrix of point clouds usually contains many blank pixels due to the lack of reflection information. Larger receptive fields can capture more reflection information to avoid insufficient learning. Considering that small object features are likely to be lost in deeper layers, we use dense local layers in the rest of MKRB to learn hierarchical features.

Specifically, it includes a plurality of convolution layers with the batch norm and activation layers. Then, the output of each layer connects the previous output features before forwarding to the next layer to form a dense representation. This structure adaptively captures and memorizes the information from the previous to current local features. The output of the last convolution layer adds the input of MKRB to generate another layered feature. Thus, these tensors $\mathcal{T}_{\text{dep}}$, $\mathcal{T}_{\text{coor}}$, and $\mathcal{T}_{\text{int}}$ are expressed by

$$\mathcal{T}_{\text{dep}} = \mathcal{F}_{\text{mkrb}}(M_{\text{dep}}(u,v)) \quad (2)$$

$$\mathcal{T}_{\text{coor}} = \mathcal{F}_{\text{mkrb}}(M_{\text{coor}}(u,v)) \quad (3)$$

$$\mathcal{T}_{\text{int}} = \mathcal{F}_{\text{mkrb}}(M_{\text{int}}(u,v)) \quad (4)$$

where $M_{\text{dep}}(u,v)$, $M_{\text{coo}}(u,v)$, and $M_{\text{int}}(u,v)$ represent three 2-D matrices converted from a depth map, coordinates, and intensity, respectively. $\mathcal{F}_{\text{mkrb}}$ denotes the operation of an MRRB module.

B. 3-D LiDAR Feature Extractor

As a pioneer work, the PointNet-based network [17] has achieved remarkable performance in point cloud recognition and segmentation. Based on it, a 3-D LiDAR feature extractor module is further developed, which mainly consists of a downsampling module and an upsampling module. The downsampling module contains the fastest points sampling (FPS) layer, the nearest grouping (NG) layer, a 1-D convolutional blocks group, and a 1-D max-pooling layer. Therefore, an encoding process $\mathcal{I}^L_{\text{Enc}}(\cdot)$ can be expressed by

$$\mathcal{I}^L_{\text{Enc}} = \mathcal{P}^L_{\text{Enc}}\left(\cdots \mathcal{P}^l_{\text{Enc}}\left(\mathcal{P}^{l-1}_{\text{Enc}}\left(\cdots \mathcal{P}^1_{\text{Enc}}(I)\right)\right)\right) \quad (5)$$

where the encoding module $\mathcal{P}^l_{\text{Enc}}(\cdot)$ is computed by

$$\mathcal{P}^l_{\text{Enc}} = \text{Max}\{\text{ReLu}(\text{Conv}(W^l_{\text{Enc}}, \text{Down}(\mathcal{I}^{l-1}_{\text{Enc}})))\} \quad (6)$$

where $\text{Conv}(\cdot)$ represents the convolution layer, $\text{ReLu}(\cdot)$ represents the rectifying linear nonlinearity (ReLU) layer, W^l_{Enc} denotes the trained weights, and $\text{Down}(\cdot)$ denotes the downsampling layer.

The upsampling block consists of a three interpolation layer and a 1-D convolutional blocks group. The decoding process $\mathcal{I}^L_{\text{Dec}}(\cdot)$ can be expressed by

$$\mathcal{I}^L_{\text{Dec}} = \mathcal{P}^L_{\text{Dec}}(\cdots \mathcal{P}^l_{\text{Dec}}(\mathcal{P}^{l-1}_{\text{Dec}}(\cdots \mathcal{P}^1_{\text{Dec}}(\mathcal{I}^L_{\text{Enc}})))). \quad (7)$$

$\mathcal{P}^l_{\text{Dec}}(\cdot)$ represents a decoding module, which is computed by

$$\mathcal{P}^l_{\text{Dec}} = \text{ReLu}(\text{Conv}(W^l_{\text{Dec}}, Up\{\mathcal{I}^{l-1}_{\text{Dec}}, \mathcal{I}^{L-l+1}_{\text{Enc}}\})), \quad (8)$$

where $\text{Up}(\cdot)$ represents the upsampling layer.

As the number of points increases in the 64-line LiDAR sensor, we choose a larger downsampling ratio in the first downsampling block to reduce the model size. It is noted that the distribution of point clouds is similar when performing the downsampling operation. For instance, 4-thousand points can still retain the valid 3-D features as 180-thousand points.

1) Multimodal 2-D Feature Fusion: $\mathcal{F}_{\text{fusion}}$ takes the fused 2-D and 3-D features as inputs and delivers a segmentation result. We adopt three 2-D convolutional blocks as a 2-D feature extractor, and the kernel size of all convolutional blocks is 1×1

$$\mathcal{F}_{\text{fusion}} = \text{Max}\{\text{ReLu}(\text{Conv}(W^l_{\text{Enc}}, \mathcal{T}_{\text{concat}}))\} \quad (9)$$

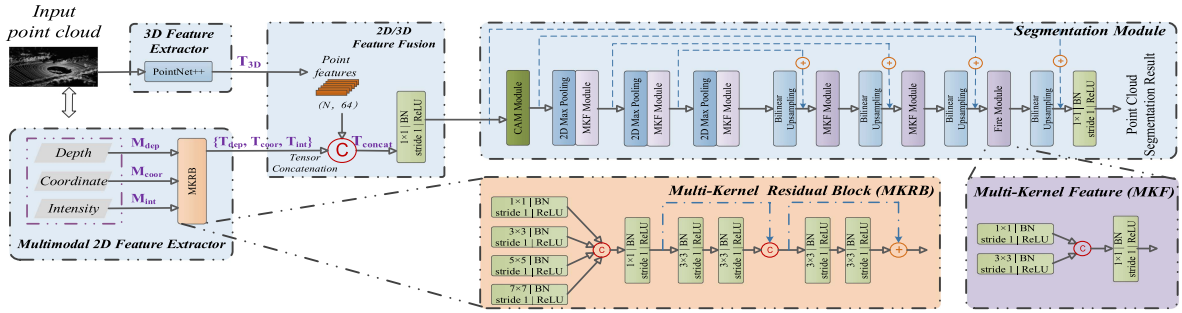


Fig. 3. Illustration of LiDAR data segmentation network based on multimodal fusion, including a multimodal 2-D feature extractor ($T_{\rm dep}$, $T_{\rm coor}$, and $T_{\rm int}$), a 3-D LiDAR feature extractor ($T_{\rm 3D}$), and a LiDAR semantic segmentation module ($T_{\rm concat} = \{T_{\rm 3D}, T_{\rm dep}, T_{\rm coor}, T_{\rm int}\}$). The MKRB is computed by (2)–(4), the fusion module is computed by (9), and the multikernel feature module is described in Section III-C.

where $\mathcal{T}_{\rm concat}$ is computed by concatenating the feature maps $\mathcal{I}^L_{\rm Dec}$, $\mathcal{T}_{\rm dep}$, $\mathcal{T}_{\rm coor}$, and $\mathcal{T}_{\rm int}$.

C. LiDAR Semantic Segmentation

Inspired by the super capability of CNN in feature extraction, we adopt an improved SegNet [26] to extract 2-D features and obtain the segmentation result $I_{\rm seg}$. Specifically, each encoder in the network is convolved with a multi-kernel filter (MKF) to generate a set of feature maps. In MKF, input tensors are fed to a parallel 1×1 kernel and a 3×3 convolution kernel to obtain multi-receptive features. It is then batch normalized and fed into a ReLU layer. Each encoder layer has a corresponding decoder layer, and the decoder output is a pixelwise 2-D feature. A skip connection is added between each encoder and decoder layer, through which shallow convolutional layer features can be introduced. In the last layer, the 1×1 convolutional kernel is utilized to obtain the pixelwise segmentation result $I_{\rm seg}$. The architecture of our multimodal fusion-based LiDAR data segmentation network is illustrated in Fig. 3.

V. Proposed Hybrid Coding of LiDAR Point Clouds

After performing the proposed multimodal fusion-based segmentation, LiDAR point data can be expressed as a combination of several independent disjoint regions $I_{\rm seg} = \{I_{\rm movable}, I_{\rm ground}, I_{\rm other}\}$. As a result, pedestrian and vehicle points, denoted by $I_{\rm movable}$, can be separated. For ground points $I_{\rm ground}$, we use a plane fitting to establish an ideal plane and encode the residuals between the real plane and the ideal plane to remove the spatial redundancy. For the other points $I_{\rm other}$, we design a prediction network to remove the temporal redundancy between the current point clouds and the previously decoded ones. The architecture of our hybrid LiDAR coding framework is illustrated in Fig. 4.

A. Surface Fitting for the Ground Points

1) Plane Model: In this section, we assume that LiDAR point data is collected in structured scenes (e.g., campus, city, road, residential, etc.) of automatic driving, and the ground points can be fitted by a plane model. Let $\hat{\mathbf{p}}_{x,y,z}$ denote a point on a fitted plane, and let $\mathbf{n}_{x,y,z}$ represent a unit vector of the fitted plane.

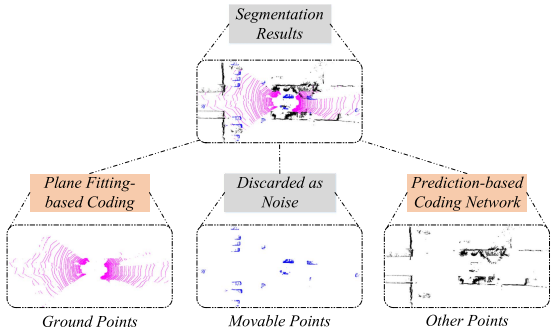


Fig. 4. Overview of the proposed hybrid coding framework for LiDAR point clouds. Based on the proposed segmentation results, point cloud data are divided into three parts: 1) ground points $I_{\rm ground}$, 2) movable points (pedestrians and vehicles) $I_{\rm movable}$, and 3) other points $I_{\rm other}$. The LiDAR data are a combination of these three disjoint sets $I = I_{\rm movable} \cup I_{\rm ground} \cup I_{\rm other}$. $I_{\rm movable}$ will be discarded before compression, $I_{\rm ground}$ is encoded by a plane fitting method, and $I_{\rm other}$ is encoded by a prediction-based coding network. The residual data $R_{\rm ground}$ and $R_{\rm other}$ will be encoded by a lossless coding scheme.

Then, the objective function of this fitted plane can be expressed by

$$\mathcal{J}(\hat{\mathbf{p}}_{x,y,z}, \mathbf{n}_{x,y,z}) = \sum_{x,y,z} [\mathbf{n}_{x,y,z} \cdot (\hat{\mathbf{p}}_{x,y,z} - \mathbf{p}_{x,y,z})]^2 \quad (10)$$

where the distance from a point to the fitted plane is calculated by $\mathbf{n}_{x,y,z} \cdot (\hat{\mathbf{p}}_{x,y,z} - \mathbf{p}_{x,y,z})$.

We use the *Levenberg–Marquardt* [27] algorithm to find surface parameters and select the best fitted plane for each region. In the global coordinate system, the fitted point cloud can be calculated from the surface equation, LiDAR sensor center coordinates, and LiDAR parameters. The difference between the actual ground points $I_{\rm ground}$ and the fitted point clouds $\hat{I}_{\rm ground}$ is computed as the residual data $R_{\rm ground}$, with which the spatial redundancy can be removed by

$$R_{\rm ground} = I_{\rm ground} - \hat{I}_{\rm ground}. \quad (11)$$

B. End-to-End Prediction Network

Usually, in real scenes, consecutive point cloud frames share a large number of overlapping regions, and the idea of the prediction network is to explore the temporal redundancies. Given an

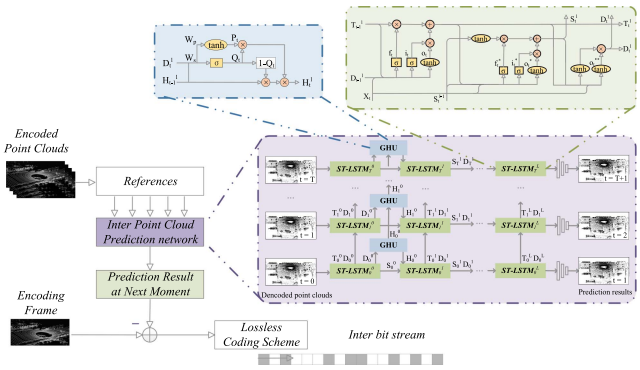


Fig. 5. Illustration of the proposed unsupervised prediction network for LiDAR data. It consists of several stacked ST-LSTM modules and GHU modules. GHU is utilized to prevent the vanishing of long-term gradients. According to the decoded frames $I_{t_0}, \ldots, I_{t_T}$, this prediction network forecasts the future frame $I_{t_{T+1}}$.

encoded frame I_t of 3-D point clouds, an interprediction module aims to generate a future point cloud frame at the time $t+1$. The difference between the real frame and the predicted frame will be encoded for the entropy coding.

1) Prediction Module: The proposed prediction module is illustrated in Fig. 5, which consists of a series of enhanced LSTM modules represented by ST-LSTM [28]. ST-LSTM is constructed by connecting temporal and spatial memories in a cascade manner through a gated structure. By adding more nonlinear layers in periodic transitions and increasing the depth of a network from one state to another, ST-LSTM has a stronger feature extraction ability than LSTM in representing temporal information. The lth ST-LSTM module can be expressed by

$$\mathcal{S}_t^l = f_t^* \odot \tanh(W_3 * S_t^{l-1}) + i_t^* \odot o_t^* \tag{12}$$

$$\mathcal{T}_t^l = f_t \odot T_{t-1}^l + i_t \odot o_t \tag{13}$$

$$\mathcal{D}_t^l = o_t^{**} \odot \tanh(W_5 * [\mathcal{T}_t^l, \mathcal{S}_t^l]) \tag{14}$$

where $*$, $\odot$, and σ represent the convolution, elementwise multiplication, and elementwise Sigmoid function, respectively. W_3 and W_5 denote the trained weights of 1×1 convolution filters. The dual-storage state jointly determines the final output.

Due to the fact that the recursion depth increases significantly along the spatial–temporal transition path, the cascade memory outperforms the simple concatenated structure of LSTMs. Each pixel in the resulting frame will have a larger input receptive field at each time step, which provides the prediction model with better short-term dynamics and abrupt changes.

In addition to short-term dynamic changes, ST-LSTM has long suffered from the gradient back-propagation. In particular, the temporal memory may forget the occurrence of outdated frames due to long transitions. Such a loop architecture remains unresolved, especially for the periodic motion or frequent occlusion. We need an information highway to learn the frame skip relationship. A gradient highway unit (GHU) is utilized to prevent the rapid disappearance of long-term gradients. The kth GHU module is defined by

$$\mathcal{Q}_t^k = \sigma(W_{sd} * \mathcal{D}_t^k + W_{sh} * \mathcal{H}_{t-1}^k) \tag{15}$$

$$\mathcal{P}_t^k = \tanh(W_{pd} * \mathcal{D}_t^k + W_{ph} * \mathcal{H}_{t-1}^k) \tag{16}$$

$$\mathcal{H}_t^k = \mathcal{P}_t \odot Q_t + (1 - Q_t) \odot \mathcal{H}_{t-1}^k \tag{17}$$

where $W_{sd,sh,pd,ph}$ denote the convolutional filters, $\mathcal{P}_t$ denotes a switch gate that can learn the transformation between $\mathcal{D}_t^k$ and the hidden state $\mathcal{H}_{t-1}^k$, and $\mathcal{H}_t^k$ represents the input for the next layer.

2) Loss Function: The proposed prediction module is represented by $\mathcal{F}_p(I_{\text{other}(t=0\ldots T)}, \vartheta)$. In other words

$$\hat{I}_{\text{other}_{(T+1)}} = \mathcal{F}_p(I_{\text{other}(t=0,\ldots,T)}; \vartheta) \tag{18}$$

where $\hat{I}_{\text{other}_{(T+1)}}$ denotes a predicted point cloud frame at $t = T+1$, and ϑ denotes the relevant learned parameters.

The difference between the original frame $I_{\text{other}_{(T+1)}}$ and the predicted frame $\hat{I}_{\text{other}_{(T+1)}}$ is used to calculate the residual $R_{\text{other}_{(T+1)}}$

$$\begin{aligned} R_{\text{other}_{(T+1)}} &= I_{\text{other}_{(T+1)}} - \hat{I}_{\text{other}_{(T+1)}} \\ &= I_{\text{other}_{(T+1)}} - \mathcal{F}_p(I_{\text{other}(t=0,\ldots,T)}; \vartheta) \end{aligned} \tag{19}$$

Finally, we adopt the L_1-norm and L_2-norm loss to train the prediction model $\mathcal{F}_p$

$$\mathcal{L} = \left\| R_{\text{other}_{(T+1)}} \right\|_1 + \left\| R_{\text{other}_{(T+1)}} \right\|_2. \tag{20}$$

At the stage of testing, the residual data will be encoded by a lossless coding scheme.

VI. Results

In this section, we verify the performance of the proposed segmentation and hybrid coding schemes on two large-scale public datasets, including SemanticKITTI [29] and KITTI [30].

A. Datasets

1) SemanticKITTI: SemanticKITTI is a large-scale semantic segmentation dataset consisting of 21 sequences, including 43552 LiDAR scans. The point cloud sequences 00th to 05th and 07th to 10th are utilized for training, the sequence 06th for validating, and the sequence 11th to 21st (20 351 scans) for testing.

2) KITTI: To demonstrate the proposed framework, we also evaluate the compression performance of SA-LPCC on the KITTI dataset, which is the largest public LiDAR point cloud dataset in the world. The KITTI dataset has been used to evaluate a stereo image, optical flow, visual odometry, 3-D object detection, and 3-D tracking, which helps us evaluate the performance of downstream tasks. The original datasets are classified as "road," "city," "residential," and "campus." We verify the performance of SA-LPCC for these four scenarios.

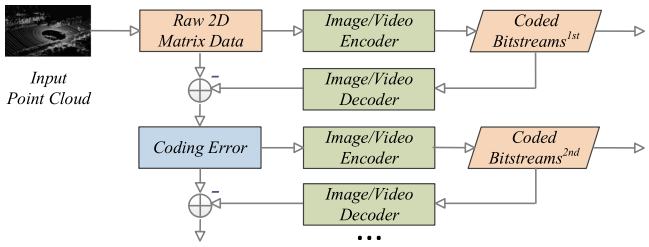


Fig. 6. Architecture for the proposed PC-HEVC method. The input LiDAR data is first converted to a 2-D matrix, quantized to 8 b, and encoded by the HEVC standard method. The coding error is calculated, and this encoding procedure is repeated until the reconstruction accuracy is met.

B. Experimental Details

1) Point Cloud Segmentation Baselines: We compare our segmentation network with some representative methods, including *SqueezeSeg* [11], *SqueezeSegV2* [12], *SqueezeSegV3* [13], *RangeNet++* [14], *FPS-Net* [15], *PolarNet* [16], and *RandLA-Net [19]*.

2) Point Cloud Coding Baselines: The point cloud compression experiments include seven latest baselines, such as *Google Draco*, *MPEG TMC13*, and tree-based point cloud compression algorithms. Image and video codec-based approaches are most effective at compressing the depth and LiDAR data. We also build two baselines, termed 1) *PC-PNG* and 2) *PC-HEVC*, to compress the LiDAR point clouds based on the PNG and high efficiency video coding (HEVC) standards, respectively. Considering that HEVC cannot directly compress a 16-b depth map [32], we propose a two-way compression: first quantizing it into 8-b image compression, and then encoding the residual data, as illustrated in Fig. 6.

3) Implementation Details: We use *TensorFlow 1.13.1*, *CUDA10.0*, and *CUDNN7.4* to implement our network. The proposed deep networks have been trained and validated on a server equipped with an *NVIDIA TITAN RTX* GPU. The AdmaOptimizer [33] is used as our optimization solver with a learning rate of $\eta = 5E-4$, $\beta = 0.99$, and $\beta_1 = 0.9$. Our model is trained over 40 000 epochs with the batch size of 8 constrained by our GPU memory.

C. Evaluation Metrics

1) Evaluation Metrics for Point Cloud Segmentation: The segmentation performance is measured in terms of Intersection over Union (IoU) and mean IoU (mIoU). IoU is calculated based on the area overlap between the prediction mask P_c and the ground-truth mask G_c. IoU is defined by

$$\mathrm{IoU}_c = \frac{P_c \cap G_c}{P_c \cup G_c} \tag{21}$$

where the subscript c denotes a segmentation class.

In addition, mIoU denotes the average of IoU over all N classes, and is defined by

$$\mathrm{mIoU} = \frac{1}{N}\sum_{c=1}^{N} \mathrm{IoU}_c. \tag{22}$$

2) Evaluation Metrics for Point Cloud Compression: The quality of LiDAR data coding is a tradeoff between compression ratio and reconstruction quality. To assess the performance, we consider the calculation of the compression rate (CR), the average bits per point (BPP), and the symmetric point cloud distance (D_d). CR is obtained by calculating the ratio between the size of the compressed data and that of the original one. The lower the value is, the better the performance. CR is defined by

$$\mathrm{CR} = \frac{\mathrm{Compressed}_{\mathrm{size}}}{\mathrm{Original}_{\mathrm{size}}} \times 100\%, \tag{23}$$

where CR denotes the compression rate. $\mathrm{Original}_{\mathrm{size}}$ and $\mathrm{Compressed}_{\mathrm{size}}$ represent the size of the point cloud data before and after compression, respectively.

Let D_d represent the distance between a raw point cloud I_{input} and the related reconstructed one I_{decode}, which is defined by

$$D_d(I_{\mathrm{input}}, I_{\mathrm{decode}}) = \frac{\overline{D}_d(I_{\mathrm{input}}, I_{\mathrm{decode}})}{2} + \frac{\overline{D}_d(I_{\mathrm{decode}}, I_{\mathrm{input}})}{2}. \tag{24}$$

$\overline{D}_d(I_i, I_j)$ denotes the average distance between point I_i and point I_j, which is defined by

$$\overline{D}_d(I_i, I_j) = \frac{1}{\|I_i\|}\sum\nolimits_{\mathbf{p}_i \in I_i} \min_{\mathbf{p}_j \in I_j} \left\|\mathbf{p}_i - \mathbf{p}_j\right\|_2^2 \tag{25}$$

where $\|I_i\|$ denotes the total number of point in I_i.

The metric in (24) is sensitive to false positives (e.g., rebuilding points in unoccupied areas) and false negatives (e.g., excluding occupied areas). For each point in I_i, we take the distance to the nearest point in I_j, and vice versa. The *Euclidean* distance is efficiently computed via a *k*-d tree in the L_2 norm.

D. Segmentation Performance

1) Quantitative Results: On the SemanticKITTI benchmark dataset, we have compared our method with representative projection-based segmentation methods. The experimental results are provided in Table I. It can be seen that the classic projection-based methods, including *SqueezeSeg* and *SqueezeSegV2*, have achieved high computational efficiency, but they have greatly sacrificed segmentation accuracy. In contrast, the latest *RangeNet++*, *PolarNet*, and *SqueezeSegV3* have achieved higher accuracy at the cost of high computational complexity. As a comparison, our method achieves the best segmentation accuracy, outperforming *RandLA-Net* (+7.35%) and *SqueezeSegV3*(+5.36%). The main reasons can be that it not only uses the spatial structure features of 3-D point clouds, but also utilizes 2-D texture features, thereby improving the segmentation accuracy of objects, such as trucks, motorcycles, bicyclists, and motorbicyclists.

2) Qualitative Results: The segmentation results of several LiDAR data samples are illustrated in Fig. 7. Specifically, Fig. 7(c) and (f) shows the segmentation results of two point clouds from the semanticKITTI dataset. Fig. 7(b) and (e) shows the corresponding ground-truths. As seen, the segmentation network can correctly identify most points, especially for roads, sidewalks, cars, and other categories. On the other hand, segmentation errors are prone to occur near object transition regions,

TABLE I
QUANTITATIVE RESULTS OF EIGHT DIFFERENT APPROACHES ON THE SEMANTICKITTI DATASET

Method	FPS	mIoU(%)	Movable Points								Ground Points			
			car	truck	bicycle	motorcycle	other-vehicle	person	bicyclist	motorbicyclist	road	sidewalk	parking	other-ground
PointNet [16]	14.6	14.92	61.6	0.1	1.3	0.3	0.8	0.2	0.2	0.0	61.6	35.7	15.8	1.4
PointNet++ [30]	20.1	16.41	53.7	0.9	1.9	0.2	0.2	0.9	1.0	0.0	72.0	41.8	18.7	5.6
SqueezeSeg [10]	72.6	24.48	68.8	3.3	16.0	4.1	3.6	12.9	13.1	0.9	85.4	54.3	26.9	4.5
SqueezeSegV2 [11]	62.8	34.53	81.8	13.4	18.5	17.9	14.0	20.1	25.1	3.9	88.6	67.6	45.8	17.7
SqueezeSegV3 [12]	5.8	49.88	92.5	29.6	38.7	36.5	33.0	45.6	46.2	20.1	91.7	74.8	63.4	26.4
RangeNet++ [13]	14.9	45.16	91.4	25.7	25.7	34.4	23.0	38.3	38.8	4.8	91.8	75.2	65.0	27.8
FPS-Net [14]	20.8	47.67	91.1	37.1	48.6	37.8	30.0	60.5	57.8	7.5	91.1	74.6	61.9	26.0
PolarNet [15]	15.6	46.10	93.8	22.9	40.3	30.1	28.5	43.2	40.2	5.6	90.8	74.4	61.7	21.7
RandLA-Net [18]	15.1	47.89	94.2	40.1	26.0	25.8	38.9	49.2	48.2	7.2	90.7	73.7	60.3	20.4
SA-LPCC	4.76	55.24	92.4	68.8	18.3	55.9	57.7	22.8	65.8	65.3	89.5	57.5	39.0	29.9

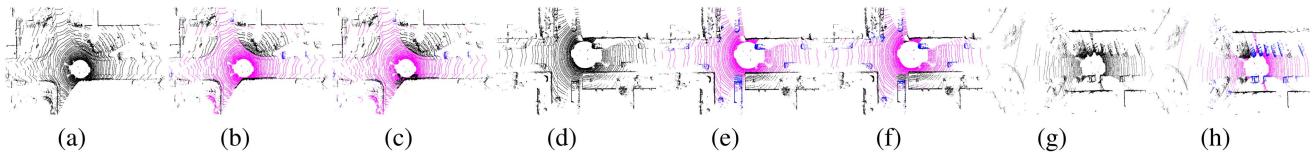


Fig. 7. Qualitative segmentation results, where pink denotes ground points, blue denotes movable points, and black denotes other points. (a) Input point clouds #5(27). (b) Ground truth #5(27). (c) Segmentation result #5(27). (d) Input point clouds #7(0). (e) Ground truth #7(0). (f) Segmentation result #7(0). (g) Input point clouds #13(9). (h) Segmentation results #13(9) (Best viewed by zoom-in).

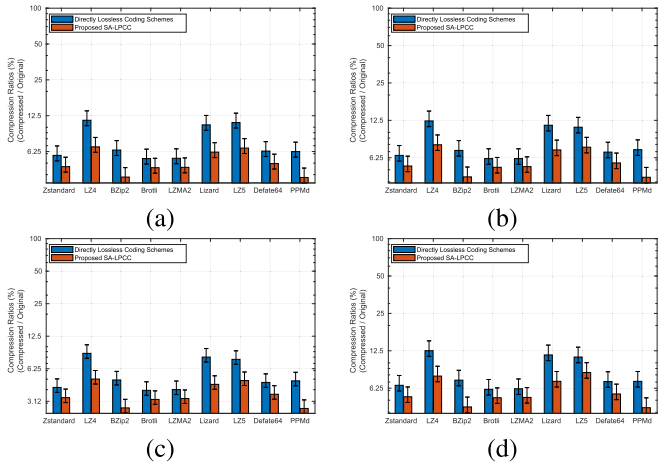


Fig. 8. Compression ratio with lossless coding schemes. (a) Sequence #11. (b) Sequence #12. (c) Sequence #13. (d) Sequence #14 (Best viewed by zoom-in).

making point cloud segmentation a more challenging task. Note that an accurate semantic segmentation can improve the performance of the subsequent compression.

E. Compression Performance

1) Compression With Lossless Coding Schemes: To find the most efficient coding method for the residual data, several lossless coding schemes are utilized to encode residual data, including *Zstandard*, *LZ4*, *BZip2*, *Brotli*, *LZMA2*, *Lizard*, *LZ5*, *Deflate64*, and *PPMd*. For comparison, we also directly compress the LiDAR data using these lossless compression schemes. Experimental results are provided in Fig. 8.

TABLE II
COMPRESSION RATIO RESULTS BETWEEN SA-LPCC AND *OCTREE*

Sequence	SA-LPCC Quantization Accuracy			Octree [36] Distance Resolution		
	1mm	5mm	1cm	$1mm^3$	$5mm^3$	$1cm^3$
#11	4.21	3.25	2.85	30.01	19.12	9.31
#12	4.74	3.73	3.26	31.01	20.43	11.53
#13	3.66	2.84	2.45	33.56	25.93	8.92
#14	4.51	3.54	3.10	30.66	20.08	9.40
#15	4.30	3.36	2.95	34.18	26.45	9.88
#16	3.96	3.00	2.58	31.28	20.53	9.89
#17	4.61	4.24	3.16	30.49	20.03	11.18
#18	3.96	3.40	2.53	29.67	18.84	9.07
#19	3.86	3.24	2.44	31.67	20.93	10.48
#20	3.81	3.04	2.45	28.43	18.71	10.21
#21	4.20	3.54	2.74	26.39	15.29	10.56
Average	4.16	3.38	2.77	30.66	20.57	10.03

The input data are the same clean LiDAR Map.

It can be observed that our method achieves a lower compression ratio compared with the other methods, which proves that SA-LPCC efficiently reduces the redundancy of LiDAR data. Meanwhile, the best coding performance is obtained when using *PPMd*. In the subsequent experiments, we choose the *PPMd* scheme as a representative of the lossless compression.

2) Comparison With Octree-Based Method: Table II provides the compression results of our SA-LPCC compared with the *Octree* method [37]. We have encoded the 10th to 99th frames of each sequence and calculated the compression ratio based on the averaged results. In the experiments, the input data of Octree are the same as that of SA-LPCC. In general, the smaller the compression ratio, the smaller the space required for data storage and the better the coding performance. The encoding accuracy of *Octree* is set to 1 mm^3, 5 mm^3, and 1 cm^3. The quantization accuracy is 1-mm, 5-mm, and 5-cm, respectively.

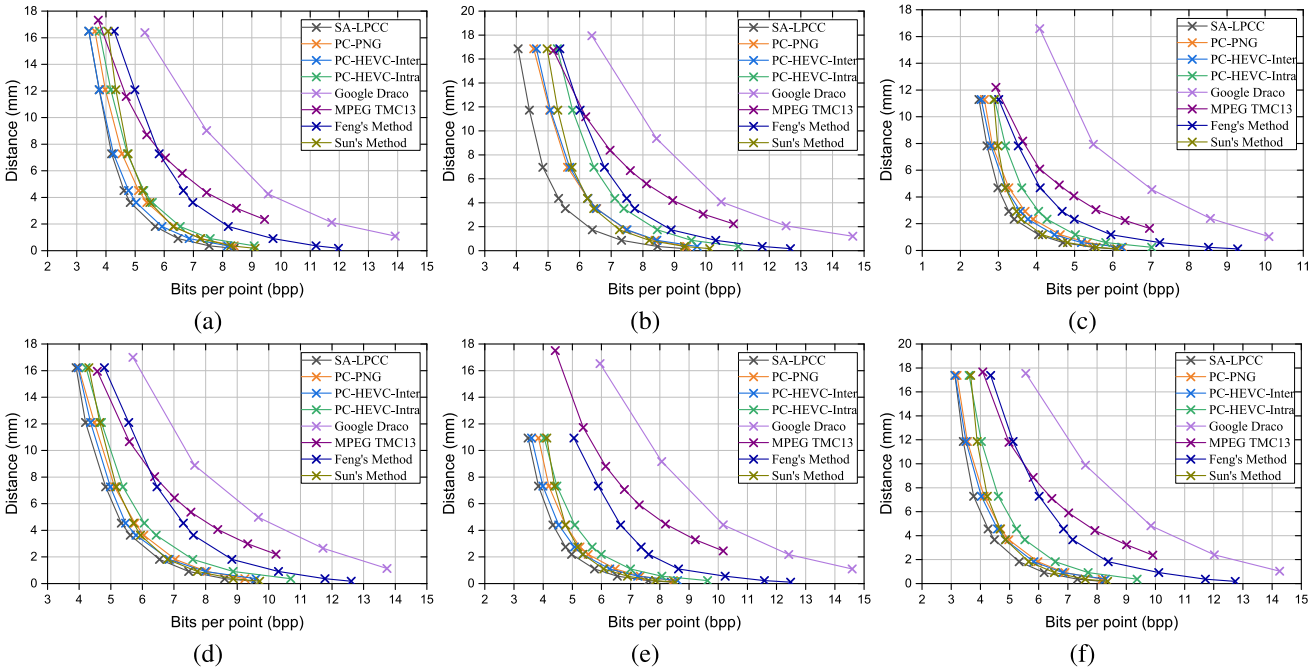


Fig. 9. Distance-bpp curves of our method in comparison with *PNG*, *HEVC* [34], *Google Draco* [22], *MPEG TMC13* [35], *Feng's* method [24], and *Sun's* method [36]. (a) Sequence #11. (b) Sequence #12. (c) Sequence #13. (d) Sequence #14. (e) Sequence #15. (f) Sequence #16 (Best viewed by zoom-in).

TABLE III
AVERAGE COMPLEXITY COMPARISONS OF SA-LPCC VERSUS *OCTREE*, *GOOGLE DRACO*, *MPEG TMC13*, *FENG'S* METHOD, AND *SUN'S* METHOD(S)

Time	SA-LPCC	PC-PNG	PC-HEVC-Intra	PC-HEVC-Inter	Octree	Draco	TMC13	Feng's Method	Sun's Method
Encoding	1.715	0.106	2.054	8.489	0.024	0.031	0.649	0.019	27.12
Decoding	1.160	0.036	0.038	0.037	0.009	0.010	0.337	0.019	0.04

As seen, compared with *Octree*, the proposed SA-LPCC obtains a smaller compression ratio.

3) Rate-Distortion Curves: Fig. 9 illustrates the rate-distortion curves of our SA-LPCC in comparison with *Google Draco* [22], *MPEG TMC13* [35], Video-based method [34], Image-based methods (PNG and HEVC), *Feng's* method [24], and *Sun's* method [36]. D_d − bpp curves reflect the relationship between distortion and compression ratio. Smaller bpp and D_d mean better coding performance, because it can achieve lower distortion with smaller bandwidth. As seen, the proposed method yields better performance among these methods. Noted that *Google Draco* and *MPEG TMC13* are not specifically designed for LiDAR data, so they cannot effectively remove redundancy. Our SA-LPCC removes the spatial redundancy of LiDAR data by plane fitting and removes the temporal redundancy by a deep prediction network. It not only considers the spatial structure characteristics of LiDAR data but also utilizes data-driven technology. Compared with *PC-PNG*, *PC-HEVC-Intra*, and *PC-HEVC-Inter*, the proposed SA-LPCC achieves smaller distortion results at the same bpp.

4) Computational Complexity: Table III provides the average encoding and decoding time of the proposed method compared with *Octree*, *Google Draco*, *MPEG TMC13*, *PC-PNG*, *PC-HEVC-Intra*, *PC-HEVC-Inter*, *Feng's* method, and *Sun's* method. It can be seen that *Octree*, *Google Draco,* and *Feng's* scheme have lower complexity, but they also show lower compression performance. *PC-HEVC-Inter* has a higher data compression ratio but higher complexity. In contrast, our SA-LPCC has a higher data compression ratio and lower complexity than *PC-HEVC-Inter*.

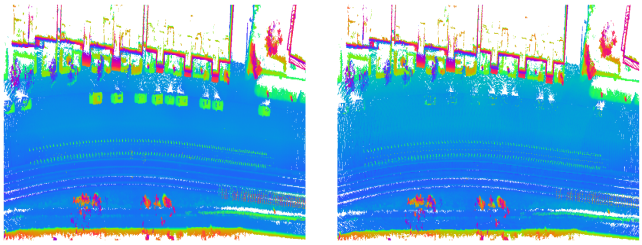
TABLE IV
COMPUTATIONAL COMPLEXITY OF THE ENTIRE PIPELINE(S)

Sequence	Pre-Processing		Encoder			Decoder
	Semantic Segmentation	Data Processing	Surface Fitting	Inference Network	Entropy Coding	
#11	0.171	1.421	0.253	0.374	1.096	1.152
#12	0.172	1.416	0.274	0.342	1.107	1.163
#13	0.174	1.345	0.264	0.343	1.078	1.158
#14	0.190	1.362	0.302	0.340	1.104	1.153
#15	0.169	1.411	0.257	0.343	1.111	1.156
#16	0.176	1.439	0.269	0.340	1.093	1.176
Average	0.175	1.399	0.270	0.347	1.098	1.160

Taking the preprocessing process into account, the computational complexity of our entire pipeline is provided in Table IV. The time cost spent on the preprocessing module and the encoder module is 1.574 s/frame and 1.715 s/frame, respectively. Experimental results show that future studies need to focus on

TABLE V
REGISTRATION ERROR UNDER DYNAMIC ENVIRONMENT

Sequence 20		**Raw Data** (w/o)		**Raw Data**		SA-LPCC (w/o)	
Source	Target	θ_{ie}	t_{ie}	θ_{ie}	t_{ie}	θ_{ie}	t_{ie}
560	561	0.0716	0.0034	0.0742	0.0286	0.0741	0.0034
562	563	0.0409	0.0013	0.1400	0.0385	0.0700	0.0044
564	565	0.0432	0.0041	0.1139	0.0311	0.0563	0.0140



(a) (b)

Fig. 10. Mapping results on the KITTI dataset. (a) Using the original point clouds. (b) Using the decoded point clouds with SA-LPCC. As seen, there are vehicles in the point cloud map built using raw point clouds, which may have confrontational impacts when used for localization or navigation. Using our SA-LPCC, the "noise" information is effectively removed, and a clean map is obtained.

further optimizing coding algorithms to reduce the computational complexity.

F. Performance on Downstream Tasks

1) Localization Performance: To further evaluate the positioning performance, the matching accuracy metric of a model is defined [38]. For example, the point cloud rotation error θ_{ie} is formulated as follows:

$$\theta_{ie} = \cos^{-1}\left(\frac{\text{trace}(R_i R_{iGT}^{-1}) - 1}{2}\right)\frac{180}{\pi} \tag{26}$$

where R_{iGT} represents a ground-truth rotation, and R_i represents a rotation calculated at each time. The point cloud translation error t_{ie} is defined as follows:

$$t_{ie} = \frac{||t_i - t_{iGT}||}{d_{\text{res}}} \tag{27}$$

where t_{iGT} represents a ground-truth translation, and t_i represents a calculated translation at each time. d_{res} represents the resolution, which normalizes the difference between translation vectors and is scale-independent.

We randomly select 6 frames from sequence #20 to test the registration performance using the iterative closest point (ICP) algorithm [38]. Sequence 20 is collected in a dynamic scene fulling of moving vehicles. The number of iterations and d_{res} is set to 200 and 1, respectively. It can be observed from Table V that our SA-LPCC has a smaller matching error after removing the dynamic objects (denoted by "w/o"), where CR $= \frac{1}{29}$.

2) Mapping Performance: Fig. 10 shows the construction results of LiDAR data, including our SA-LPCC decoded point clouds and the original point clouds. The decoded LiDAR map will serve as a prior for autonomous driving, which needs a clean map (e.g., no pedestrians and no vehicles). The main reason is that these objects will reduce the positioning accuracy and affect the path planning, and these points will be treated as obstacles. It can be observed that there are vehicles built with the raw point clouds. In contrast, our SA-LPCC method can effectively address this problem and obtain clean maps that the way for the subsequent localization and navigation. In summary, the proposed SA-LPCC considers the specific task requirements of unmanned driving and removes the obstacle points from the source, which can significantly save the bandwidth and storage space.

VII. CONCLUSION

In this article, we introduce a task-driven scene-aware LiDAR point cloud framework for autonomous driving. To be more specific, we develop a multimodal-based semantic segmentation scheme to eliminate the influence of movable objects in LiDAR data. Considering the spatial structure features of LiDAR data, we develop a plane-fitting-based method to remove the spatial redundancy. Based on the characteristics of the structural similarities between adjacent point clouds, we investigate a prediction-based end-to-end unsupervised interframe compression network. Experimental results show that our SA-LPCC method can compress a LiDAR map to nearly 1/30 of its original size with the millimeter-level accuracy, paving the way for the subsequent localization and navigation applications.

In addition, our SA-LPCC is an early LiDAR-coding-based scene-aware scheme for autonomous driving. There are some potentially important issues that need to be addressed in the future. First, to reduce the computational complexity of SA-LPCC, we plan to further investigate some specific optimization approaches for deep models, including knowledge distillation, weight quantization, pruning, and network structure optimization. Second, due to the fact that there are holes after removing the relevant objects, follow-up research will focus on finding a balance between repairing accuracy [39], [40] and compression efficiency. Third, brushes and tree leaves may also introduce noise to the LiDAR point data registration in some cases, which has little impact on the safety of autonomous driving. Since brushes and leaves do not have fixed shapes, some effective segmentation investigations can be explored to filter them out, which will further improve compression efficiency.

REFERENCES

[1] M. Aldibaja, N. Suganuma, and K. Yoneda, "Robust intensity-based localization method for autonomous driving on snow–wet road surface," *IEEE Trans. Ind. Informat.*, vol. 13, no. 5, pp. 2369–2378, Oct. 2017.

[2] J. Li, H. Qin, J. Wang, and J. Li, "OpenStreetMap-based autonomous navigation for the four wheel-legged robot via 3D-LiDAR and CCD camera," *IEEE Trans. Ind. Electron.*, vol. 69, no. 3, pp. 2708–2717, Mar. 2022.

[3] L. Wen and K.-H. Jo, "Three-attention mechanisms for one-stage 3D object detection based on LiDAR and camera," *IEEE Trans. Ind. Informat.*, vol. 17, no. 10, pp. 6655–6663, Oct. 2021.

[4] A. Dolatabadi, H. H. Abdeltawab, and Y. A.-R. Mohamed, "Deep spatial-temporal 2-D CNN-BLSTM model for ultra-short-term LiDAR-Assisted wind turbine's power and fatigue load forecasting," *IEEE Trans. Ind. Informat.*, vol. 18, no. 4, pp. 2342–2353, Apr. 2022.

[5] W. Xie, Y. Wang, H. Chen, and D. D.-U. Li, "128-channel high-linearity resolution-adjustable time-to-digital converters for LiDAR applications: Software predictions and hardware implementations," *IEEE Trans. Ind. Electron.*, vol. 69, no. 4, pp. 4264–4274, Apr. 2022.
[6] A. Varischio, F. Mandruzzato, M. Bullo, M. Giordani, P. Testolina, and M. Zorzi, "Hybrid point cloud semantic compression for automotive sensors: A performance evaluation," in *Proc. IEEE Int. Conf. Commun.*, 2021, pp. 1–6.
[7] X. Sun, S. Wang, M. Wang, Z. Wang, and M. Liu, "A novel coding architecture for LiDAR point cloud sequence," *IEEE Robot. Automat. Lett.*, vol. 5, no. 4, pp. 5637–5644, Oct. 2020.
[8] Q. Zhou, Z. Shen, B. Yong, R. Zhao, and P. Zhi, *Theories and Practices of Self-Driving Vehicles*. Amsterdam, The Netherlands: Elsevier, 2022.
[9] H. Gao, B. Cheng, J. Wang, K. Li, J. Zhao, and D. Li, "Object classification using CNN-based fusion of vision and LIDAR in autonomous vehicle environment," *IEEE Trans. Ind. Informat.*, vol. 14, no. 9, pp. 4224–4231, Sep. 2018.
[10] F. Cao, F. Yan, S. Wang, Y. Zhuang, and W. Wang, "Season-invariant and viewpoint-tolerant LiDAR place recognition in GPS-Denied environments," *IEEE Trans. Ind. Electron.*, vol. 68, no. 1, pp. 563–574, Jan. 2021.
[11] B. Wu, A. Wan, X. Yue, and K. Keutzer, "SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR point cloud," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2018, pp. 1887–1893.
[12] B. Wu, X. Zhou, S. Zhao, X. Yue, and K. Keutzer, "SqueezeSegV2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a LiDAR point cloud," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 4376–4382.
[13] C. Xu et al., "SqueezeSegV3: Spatially-adaptive convolution for efficient point-cloud segmentation," in *Proc. Eur. Conf. Comput. Vis.*, 2020, pp. 1–19.
[14] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet : Fast and accurate LiDAR semantic segmentation," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2019, pp. 4213–4220.
[15] A. Xiao, X. Yang, S. Lu, D. Guan, and J. Huang, "FPS-Net: A convolutional fusion network for large-scale LiDAR point cloud segmentation," *J. Photogrammetry Remote Sens.*, vol. 176, pp. 237–249, 2021.
[16] Y. Zhang et al., "PolarNet: An improved grid representation for online LiDAR point clouds semantic segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 9601–9610.
[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "PointNet: Deep learning on point sets for 3D classification and segmentation," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2017, pp. 652–660.
[18] C. Zhao, W. Zhou, L. Lu, and Q. Zhao, "Pooling scores of neighboring points for improved 3D point cloud segmentation," in *Proc. IEEE Int. Conf. Image Process.*, 2019, pp. 1475–1479.
[19] Q. Hu et al., "RandLA-Net: Efficient semantic segmentation of large-scale point clouds," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit.*, 2020, pp. 11105–11114.
[20] R. L. de Queiroz, D. C. Garcia, P. A. Chou, and D. A. Florencio, "Distance-based probability model for Octree coding," *IEEE Signal Process. Lett.*, vol. 25, no. 6, pp. 739–742, Jun. 2018.
[21] M. Krivokuća, P. A. Chou, and M. Koroteev, "A volumetric approach to point cloud compression—Part II: Geometry compression," *IEEE Trans. Image Process.*, vol. 29, pp. 2217–2229, Dec. 11, 2019.
[22] Google, "Draco: 3D data compression," 2022. [Online]. Available: https://github.com/google/draco
[23] C. Tu, E. Takeuchi, A. Carballo, C. Miyajima, and K. Takeda, "Motion analysis and performance improved method for 3D LiDAR sensor data compression," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 1, pp. 243–256, Jan. 2021.
[24] Y. Feng, S. Liu, and Y. Zhu, "Real-time spatio-temporal LiDAR point cloud compression," in *Proc. IEEE Int. Conf. Intell. Robots Syst.*, 2020, pp. 10766–10773.
[25] H. W. Yoo, R. Riegler, D. Brunner, S. G. Albert, T. Thurner, and G. Schitter, "Experimental evaluation of vibration influence on a resonant MEMS scanning system for automotive LiDARs," *IEEE Trans. Ind. Electron.*, vol. 69, no. 3, pp. 3099–3108, Mar. 2022.
[26] V. Badrinarayanan, A. Kendall, and R. Cipolla, "SegNet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 39, no. 12, pp. 2481–2495, Dec. 2017.
[27] B. M. Wilamowski and H. Yu, "Improved computation for Levenberg–Marquardt training," *IEEE Trans. Neural Netw.*, vol. 21, no. 6, pp. 930–937, Jun. 2010.
[28] J. Liu, A. Shahroudy, D. Xu, and G. Wang, "Spatio-temporal LSTM with trust gates for 3D human action recognition," in *Proc. Eur. Conf. Comput. Vis.*, 2016, pp. 816–833.
[29] J. Behley et al., "SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2019, pp. 9297–9307.
[30] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The KITTI dataset," *Int. J. Robot. Res.*, vol. 32, no. 11, pp. 1231–1237, 2013.
[31] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "PointNet++: Deep hierarchical feature learning on point sets in a metric space," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, vol. 30, pp. 5105–5114.
[32] M. Wang, W. Xie, J. Zhang, and J. Qin, "Industrial applications of ultrahigh definition video coding with an optimized supersample adaptive offset framework," *IEEE Trans. Ind. Informat.*, vol. 16, no. 12, pp. 7613–7623, Dec. 2020.
[33] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
[34] G. J. Sullivan, J. R. Ohm, W. J. Han, and T. Wiegand, "Overview of the high efficiency video coding (HEVC) standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 12, pp. 1649–1668, Dec. 2012.
[35] S. Schwarz et al., "Emerging MPEG standards for point cloud compression," *IEEE Trans. Emerg. Sel. Topics Circuits Syst.*, vol. 9, no. 1, pp. 133–148, Mar. 2019.
[36] X. Sun, Y. Sun, W. Zuo, S. S. Cheng, and M. Liu, "A novel coding scheme for large-scale point cloud sequences based on clustering and registration," *IEEE Trans. Automat. Sci. Eng.*, vol. 19, no. 3, pp. 2384–2396, Jul. 2022.
[37] A. Aldoma et al., "Point cloud library: Three-dimensional object recognition and 6 DOF pose estimation," *IEEE Robot. Automat. Mag.*, vol. 19, no. 3, pp. 80–91, Sep. 2012.
[38] H. Ye, Y. Chen, and M. Liu, "Tightly coupled 3D LiDAR inertial odometry and mapping," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2019, pp. 1–7.
[39] Y. Chen et al., "Image super-resolution reconstruction based on feature map attention mechanism," *Appl. Intell.*, vol. 51, no. 7, pp. 4367–4380, 2021.
[40] Y. Chen et al., "The improved image inpainting algorithm via encoder and similarity constraint," *Vis. Comput.*, vol. 37, no. 7, pp. 1691–1705, 2021.

Xuebin Sun received the B.S. degree from the Tianjin University of Technology, Tianjin, China, in 2011, and the M.S. and Ph.D. degrees from Tianjin University, Tianjin, in 2014 and 2018, respectively.

He is currently a Postdoctoral Research Fellow with the Guangdong Key Laboratory of Intelligent Information Processing, College of Electronics and Information Engineering, Shenzhen University, Shenzhen, China. His research interests include video coding optimization, digital image processing, and point cloud processing.

Miaohui Wang (Member, IEEE) received the Ph.D. degree from the Department of Electronic Engineering, The Chinese University of Hong Kong, Hong Kong, China.

From 2014 to 2015, he was a Researcher working on the standardization of video coding with the Innovation Laboratory, InterDigital Inc., San Diego, CA, USA. From 2015 to 2017, he was a Senior Research Staff working on computer vision and machine learning at The Creative Life (TCL) Research Institute of Hong Kong, Hong Kong, China. He is currently a tenured Associate Professor with the College of Electronics and Information Engineering, Shenzhen University (SZU), Shenzhen, China. He has authored or coauthored more than 70 peer-reviewed papers in top-tier international journals and conferences. His current research interests cover a wide range of topics related to high-dimension visual data compression and transmission, medical image analysis, computer vision, and machine learning.

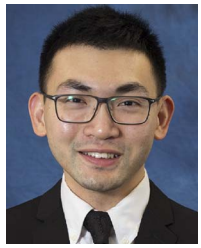
Dr. Wang was the recipient of the *Best Thesis Award* from the Ministry of Education of Shanghai City and Fudan University (FDU). He was also the recipient of the *Best Paper Award* from the *International Conference on Advanced Hybrid Information Processing* in 2018, and the *Outstanding Reviewer Award* from the *International Conference on Multimedia & Expo* in 2021.

Jingxin Du received the B.S. degree from the Chinese University of Hong Kong-Shenzhen, Shenzhen, China.

He is currently a Research Assistant with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. His research interests include mobile robots, semantic segmentation, deep learning, autonomous vehicles, etc.

Yuxiang Sun received the B.S. degree from the Hefei University of Technology, Hefei, China, in 2009, the M.S. degree from the University of Science and Technology of China, Hefei, in 2012, and the Ph.D. degree from The Chinese University of Hong Kong, Hong Kong, in 2017.

He was a Research Associate with the Department of Electronic and Computer Engineering, The Hong Kong University of Science and Technology, Hong Kong. His current research interests include autonomous driving, deep learning, robotics and autonomous systems, and semantic scene understanding.



Shing Shin Cheng (Member, IEEE) received the B.S. degree in mechanical engineering from Johns Hopkins University, Baltimore, MD, USA, in 2013, the M.S. degree in mechanical engineering from the University of Maryland, College Park, MD, in 2016, and the Ph.D. degree in robotics from the Georgia Institute of Technology, Atlanta, GA, USA, in 2018.

He is currently an Assistant Professor with the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong. His work has been published in leading robotics and materials journals and conferences. His research interests include medical robotics, image guidance and navigation, and smart sensors and actuators.

Wuyuan Xie (Member, IEEE) received the B.S. degree from Central South University, Changsha, China, in 2006, the M.S. degree from the South China University of Technology University, Guangzhou, China, in 2009, and the Ph.D. degree from the Department of Mechanical and Automation Engineering, The Chinese University of Hong Kong, Hong Kong, in 2016.

She is currently an Assistant Professor with Shenzhen University, Shenzhen, China. From 2016 to 2017, she was a Postdoctoral Fellow with The Hong Kong Polytechnic University. From 2008 to 2011, she was with the Shenzhen Institute of Advanced Technology, Chinese Academy of Science. Her research interests include image and video processing, computer vision, and machine learning.