Human Vision Based 3D Point Cloud Semantic Segmentation of Large-Scale Outdoor Scenes

Sunghwan Yoo, Yeonjeong Jeong, Maryam Jameela, Gunho Sohn
Department of Earth and Space Science and Engineering, York University
Toronto, ON M3J 1P3, Canada
(jacobyoo, yjjeong, maryumja, gsohn)@yorku.ca

Abstract

This paper proposes EyeNet, a novel semantic segmentation network for point clouds that addresses the critical yet often overlooked parameter of coverage area size. Inspired by human peripheral vision, EyeNet overcomes the limitations of conventional networks by introducing a simple but efficient multi-scale input and a parallel processing network with connection blocks between parallel streams. The proposed approach effectively addresses the challenges of dense point clouds, as demonstrated by our ablation studies and state-of-the-art performance on Large-Scale Outdoor datasets.

1. Introduction

Recently, there has been growing interest in developing digital twins of the three-dimensional world, driven by their various applications. With advancements in LiDAR devices and survey techniques, point cloud datasets have become more accurate, dense, and spatially extensive, both on the ground level [3, 14, 29, 30, 34] and in the airborne level [10, 16, 28, 35].

However, the functional coverage area of an input batch is critical for effective feature learning in semantic segmentation networks, especially as data sets become denser [16, 35]. Traditional approaches to semantic segmentation, such as increasing the number of input points [17, 33, 40] and downsampling the point cloud [25, 26] to extract features from high-density outdoor scenes, have limitations. For instance, increasing the number of input points is restricted by the memory capacity of GPUs, while further downsampling can result in information loss.

In this work, we propose a 3D semantic segmentation network called EyeNet, inspired by the human visual system, peripheral vision. The network takes input from multi-scale regions, including the central and peripheral regions. The high-density central region is advantageous for extracting precise geometric features, while the low-density peripheral region with a larger coverage area is beneficial for extracting features from objects that require such assets. To process these regions, the study uses a parallel feature processing design with connection layers for exchanging features between parallel streams. The proposed network outperforms state-of-the-art semantic segmentation methods on the SensatUrban benchmark [16] and Toronto3D [30]. The design philosophy of the EyeNet is presented in Figure 1.

Figure 1. The design of EyeNet is based on the concept of the human vision system. We drew inspiration from the peripheral vision system, which has a low density of receptors but covers a larger region than the central vision. As a result, we added the peripheral region around the central region to achieve a similar effect in our network.

Our key contributions are:

- Our approach introduces a memory-efficient human vision based input that draws inspiration from human vision, in order to address the limitations of traditional semantic segmentation networks.
- To facilitate feature learning using the human vision based input, we proposed an efficient parallel processing architecture.
- To ensure that complex features from both the central and peripheral regions are preserved and effectively
merged for improved learning, we proposed an effective deep learning connection layer.

2. Related Work

Semantic segmentation of outdoor 3D point clouds is a challenging task that has received increasing attention recently. Numerous deep learning-based approaches have been proposed to address this problem. [13, 17, 25, 26, 33, 36, 40]

Deep learning-based methods have shown promising results in semantic segmentation of outdoor 3D point clouds. However, achieving optimal coverage area of the deep learning network is one of crucial factors. Two commonly used methods to achieve an effective coverage area in outdoor 3D point clouds are downsampling the input point cloud and increasing the number of input points.

2.1. Downsampling Input Point Cloud

One of the most effective ways to manage computational costs and ensure sufficient coverage area of the input in outdoor 3D point cloud semantic segmentation is downsampling. Early networks, such as PointNet [25, 26] often utilized the constant-volume downsampling method, which collects points in a fixed volume and downsamples them to a fixed number of points. More recent networks such as KPConv [17, 33] adopted the grid sampling method where one point per voxel in a 3D grid was kept. Although this approach is computationally efficient, it disregards differences in point distribution within the point cloud, resulting in a loss of geometric information and a decrease in performance [16].

2.2. Increasing the Number of Inputs

Recently, the trend has shifted away from constant-volume downsampling towards increasing the number of input points through grid sampling [6, 17, 33, 40]. RandLA [17] is a network that processes large-scale point clouds by utilizing random sampling based on k-nearest neighbors (KNN), thus avoiding the loss of geometric information. However, a critical limitation of this method is that the ability to increase the number of input points can be restricted by the memory capacity of GPUs.

2.3. Peripheral Vision in Computer Vision

The integration of peripheral vision into deep learning has been a widely explored area in 2D computer vision research. It is primarily investigated in 2D object detection and 2D semantic segmentation [7, 9, 12, 24]. Also, recent studies have explored the combination of peripheral vision and transformer networks [23], leading to the development of the Peripheral Vision Transformer network. However, the application of peripheral vision systems in 3D computer vision tasks has not been extensively explored.

3. Methodology

3.1. Overview

Real-world 3D scene’s point clouds have tremendous points and varying densities, requiring sampling methods like grid subsampling which scale is determined by voxel size [32]. Then, unsampled points are predicted based on the nearest sampled point’s prediction usually decided by voting because random cropping is used due to GPU memory limit. When we crop the input point cloud randomly for training and inference, the optimal coverage size of the input point cloud is not well-defined.

To address this issue, we introduce a novel approach in EyeNet that utilizes both central and peripheral regions, allowing for a parallel processing network that facilitates communication between these regions to retain important features. By incorporating the human vision based input in this manner, we can maximize the benefits of these inputs. A summary of the EyeNet architecture is shown in Figure 2. Our EyeNet can basically be applied to most existing point-based networks [17, 33, 40] easily. In this paper, we select RandLA as our baseline. Therefore, any process not described in this section followed the same procedures used in the baseline RandLA implementation [1].

As depicted in Figure 2, we constructed the network by stacking two streams of RandLA-Net’s encoder and decoder in parallel. To extract fundamental features in each encoding layer, we utilized the k-NN based RandLA’s local feature aggregation (LFA) module. The LFA module outputs from both streams were randomly downsampled and fused through the connection layer. After the five encoding layers, we included five decoding layers with a bilinear interpolation operation. Skip connections were utilized to concatenate features from encoding and decoding layers to minimize information loss due to random downsampling. Finally, the outputs from both streams were merged using the Feature Merging Block and passed through fully connected layers (fc) to assign predicted labels to each point.

3.2. Human Vision Based Input

Due to the extensive number of points present in urban scale large outdoor point cloud, it is not feasible to input all points into the network simultaneously. Thus, for both training and inference purposes, it is necessary to crop the input region in the entire scene. To do this, our approach, called Human Vision Based Input, is centred around two key factors: the coverage areas and point densities of the central and peripheral regions. Let N be the number of input points for each feature processing stream. We use the K-nearest neighbours (KNN) algorithm to gather N points from the centroid $C_0$, forming the input points for the central stream $P_c = \{p_1, \ldots, p_n, \ldots, p_N\}$. Here, we define R as the distance between $C_0$ and the farthest point $p_f$ in $P_c$, and
we set the radius of the central region to $R$ and the peripheral region to $2R$. We randomly sample $N/4$ points from the central region, and we refer to these points as the messenger points, which are represented as $P_m = \{p_m^1, \ldots, p_m^n, \ldots, p_m^{N/4}\} \subseteq P_c.$

Moreover, we randomly select an additional $3N/4$ points from the region between $R$ and $2R$, and we define them as $P_{ph−m} = \{p_{ph}^1, \ldots, p_{ph}^n, \ldots, p_{ph}^{3N/4}\}.$ We choose the number $3N/4$ to ensure uniform point density with $P_m$.

Finally, the combination of the point clouds $P_{ph−m}$ and $P_m$ becomes the input point cloud for the peripheral stream, which we denote as $P_{ph} = P_{ph−m} + P_m = \{p_{ph}^1, \ldots, p_{ph}^n, \ldots, p_{ph}^{3N/4}, p_m^1, \ldots, p_m^n, \ldots, p_m^{N/4}\}.$ Therefore, the messenger points are an intersection between the points in the central and peripheral regions. An overview of the operation is shown in Figure 4.

Our proposed human vision based technique for collecting network input points allows us to cover the same input region for a deep neural network using substantially fewer points than the conventional fixed-number-based approaches like RandLA. In all experiments, we used nearly 60% fewer points than the baseline. The efficient sampling approach enabled our EyeNet to utilize GPU memory more effectively. During inference, our EyeNet was capable of employing four times more batches than the baseline due to this approach.

3.3. Parallel Processing and Receptive Field

The network comprises two streams, namely the central and peripheral streams. The central and peripheral streams process point clouds $P_c$ and $P_{ph}$, respectively, with different densities. It makes the receptive field of the peripheral region larger than that of the central region when we use the KNN-based module for feature aggregation with the same number of $K$ for both regions. In the peripheral stream, large when in the peripheral region, or both when selected as a messenger point due to the use of random cropping in large-scale 3D point cloud training and inferencing.

3.4. Connection Block

As we utilize a parallel processing network, establishing connections between two streams is crucial. We were motivated by previous works such as [18, 39] and developed the connection block that exchanges information between streams through messenger points. Since the feature information from the central and peripheral streams vary, we adopted the two-way self-attentive pooling block from [17] along with the channel enhancement block from [6] for better information exchange.

Feature Preparation: To start, we consider the features of points in the output of the central and peripheral streams, at a given layer $l$, denoted as $F_{c}^{l} = \{f_{c}^{l,1}, \ldots, f_{c}^{l,n}, \ldots, f_{c}^{l,N/4}\}$ and $F_{ph}^{l} = \{f_{ph}^{l,1}, \ldots, f_{ph}^{l,n}, \ldots, f_{ph}^{l,N/4}\},$ respectively. From these features, we extract the features of messenger points $F_{m,c}^{l} = \{f_{m,c}^{l,1}, \ldots, f_{m,c}^{l,n}, \ldots, f_{m,c}^{l,N/4}\} \subseteq F_{c}^{l}$ and $F_{m,ph}^{l} = \{f_{m,ph}^{l,1}, \ldots, f_{m,ph}^{l,n}, \ldots, f_{m,ph}^{l,N/4}\} \subseteq F_{ph}^{l}$.

The remaining points’ features are referred to as $F_{c−m} = \}$.  

Figure 2. The figure displays the architecture of EyeNet, which utilizes a parallel processing approach to handle the human vision based input. The network also includes feature connection and feature merging blocks to facilitate information exchange between the central and peripheral streams. $N$ and $d$ are used to represent the number of points and feature dimension, respectively. The central and peripheral regions share $N/4$ points as inputs, resulting in a total of $7N/4$ input points. MLP represents the multilayer perceptron.
Attention scores are denoted as follows:

\[
\begin{align*}
F_{m,c}^i & \subseteq F_c^i, \\
F_{m,ph}^i & \subseteq F_{ph}^i.
\end{align*}
\]

Afterward, we concatenate the channel-wise features of \(F_{m,c}^i\) and \(F_{m,ph}^i\) into \(F_m^i\) and process the resulting output through shared MLP function \(m(\cdot)\) three times. The shared MLPs’ operations are denoted as follows:

\[
F^i = m_3 (m_2 (m_1 (f^i_{m,ph}, W_2), W_3), W_4)
\]

where \(f^i = \{f^{i1}, \ldots, f^{in}, \ldots, f^{i3N/4}\}\) and \(W_4\) are the learnable weights of shared MLPs.

**Attention Pooling:** Inspired by self-attentive pooling block in RandLA [17], we used the powerful attention mechanism to exchange and fuse important features automatically, and it consists of the following steps.

We utilized a shared MLP followed by softmax to learn a unique attention score for each feature from \(F^i\). It is formally defined as follows:

\[
f_{ap}^i = L \left( m_a \left( f^i, W_a \right) \right)
\]

where \(W\) is the learnable weights of shared MLP and \(L\) is the Relu activation function.

\[
s_{ap}^i = S \left( f_{ap}^i \right)
\]

where \(S\) is softmax function. We can consider the attention scores that have been learned as a softmax function that automatically picks out significant characteristics. In technical terms, these traits are combined through a weighted summation process, as described below:

\[
f^{i'} = \sum (f^i \cdot s_{ap}^i)
\]

**Channel Enhancement:** In addition, we improved the features by creating more prominent global features while minimizing the loss of information during the self-attention encoding process. This method was initially presented in
RandLA++ [6] and involved the following procedures.

To start, intermediate feature sets $F^c$, $F^p$, and $F^m$ are concatenated into $F^{i, c}$. Later, the concatenated features are operated by shared MLP function $m_{ce}(\cdot)$, followed by ReLu $L$.

$$f^{i, c}_{ce} = L(m_{ce}(f^{i, c}_{ce}, W_{ce}))$$

(5)

Subsequently, the output feature $f^{i, c}_{ce}$ is improved channel-wise by attention pooling using Eq. (6) and Eq. (7) into $\hat{f}^{i, c}_{ce}$. Ultimately, a fully connected layer yields the combined feature $\hat{f}^{i, c}$. It is formally defined as follows:

$$s^{i, c}_{ce} = S(f^{i, c}_{ce})$$

(6)

$$f^{i, c}_{ce} = \sigma \left( f^{i, c}_{ce}, W_{fc} \right)$$

(7)

$$\hat{f}^{i, c}_{ce} = \sum \left( f^{i, c}_{ce} \cdot s^{i, c}_{ce} \right)$$

(8)

where $\sigma$ is a sigmoid function.

Finally, element-wise multiplication between $F^c$ and $\hat{f}^{i, c}_{ce}$ is applied, which is then added to $F^c$ to generate channel enhanced feature $F^{i, c}$.

Lastly, the outputs of both branches, $\hat{f}^{i, c}$ and $\hat{f}^{i, p}$ respectively, are added with the corresponding output of $m_1$ operation of the concatenated messenger features. Then point-wise concatenation is performed with the corresponding remaining points’ features. It is defined as follows:

$$f^{i, m, c} = \hat{f}^{i, c} + m_1(f^{i, m, c}, W_1)$$

(9)

$$f^{i, m, ph} = \hat{f}^{i, ph} + m_1(f^{i, m, ph}, W_1)$$

(10)

In summary, we begin by preparing features through our feature preparation process. These features are then divided into two streams of attention pooling and channel enhancing blocks. This is done to allow for the effective exchange and merging of features needed for both the central and peripheral streams. Please see the diagram of our connection block in Figure 3 for further illustration.

3.5. Feature Merging

The output features $\hat{F}_c$ and $\hat{F}_p$ from the central and peripheral streams, respectively, have to be merged back to match the input size of the human vision based input. We utilize squeeze and excitation method [15] to perform feature merging.

First, we extract $F^{m, c}$ and $F^{m, ph}$, features of messenger points, from $\hat{F}_c$ and $\hat{F}_p$, respectively, resulting in corresponding remained features $F^{c, m}$ and $F^{p, m}$. Next, we perform channel-wise concatenation between $F^{m, c}$ and $F^{m, ph}$ to obtain $F_m$. We then use shared MLPs to smooth the feature spaces of $F^c_m$, $F^{c, m}$, and $F^{p, m}$, followed by point-wise concatenation. Subsequently, we apply a squeeze and excitation module to enhance the salient features. The feature merging operation is illustrated in Figure 5.

4. Experiments

4.1. Implementation details

We implemented EyeNet in Tensorflow, employing the grid sampling strategy as input pre-processing, followed by KPConv [33] and RandLA [17]. The Adam optimizer is employed, with an initial learning rate of 0.005, which decays by 5% after each epoch. For each layer, the number of nearest points $K$ is set to [16, 21, 21, 21, 16]. We sample a fixed number of 28672 points from each point cloud as input, and use a batch size of 16. Our choice of loss function is lovasz-softmax loss [4]. We followed the same voxel size and input features with ones used in RandLA [17] except YUTO dataset. All experiments are performed on an
we upload the test labels on the online test server [work’s same train, validation, and test split. For evaluation, point clouds are divided into 34 tiles, and we follow their belief of 13 classes (ground, vegetation, building, etc.). The meter. Each point in this dataset is assigned a semantic label and have an average point density of 372 points per square meter. Testing our network.

Table 2. Toronto3D Performance Comparison. Performances of other works except RandLA++ are taken from the SensatUrban paper [16] RandLA++ performance is taken from its paper. [6]

NVIDIA Quadro RTX 6000 GPU.

4.2. Semantic Segmentation on Benchmark datasets

To evaluate the performance of EyeNet, we conducted experiments on the semantic segmentation task. We selected a range of outdoor benchmark datasets, including SensatUrban [16], Toronto3D [30], DALES [35], as well as our own YUTO dataset. Sensat Urban: The SensatUrban [16] dataset for the semantic segmentation task consists of large areas of three UK cities with a coverage area of 7.64 km². 2847 million points were collected using a UAV Photogrammetry and have an average point density of 372 points per square meter. Each point in this dataset is assigned a semantic label of 13 classes (ground, vegetation, building, etc.). The point clouds are divided into 34 tiles, and we follow their work’s same train, validation, and test split. For evaluation, we upload the test labels on the online test server [2] and acquire mean class-wise intersection over union (mIoU), overall point-wise accuracy (OA), and per class IoU. We set the voxel size of the grid sampling to 0.2m, and use the provided 3D coordinates and color information for training and testing our network.

The performance comparison results for SensatUrban are presented in Table 1. EyeNet has set a new state-of-the-art for both OA and mIoU metrics, surpassing all existing methods by a considerable margin. Remarkably, EyeNet has shown the best performance in seven out of thirteen semantic classes. The SensatUrban dataset is notorious for having severe class imbalance, and none of the previous approaches were able to predict the rail and bike classes, except for SPGraph [19]. In contrast, EyeNet successfully predicted these classes without any significant data engineering efforts. Furthermore, recent works such as RandLA [17], KPConv [33], Point Transformers [40], Omni-Supervised [11], FG-Net [21], and SCF-Net [8] have increased the number of input point clouds up to 100K for expanding the input coverage area. Despite using an input size of approximately 28K, which is far smaller than other approaches, EyeNet demonstrated superior performance in both small and large object classes. An illustration of segmented scenes on Sensat Urban is shown in Fig. 6.

Toronto3D: Toronto3D [30] is a publicly available dataset that contains 3D point clouds and associated images of urban environments in Toronto, Canada. The dataset covers a total length of 1 kilometer and was collected using a mobile lidar scanner. The dataset contains 78.3 million points with an average point density of 391 points per square me-

Table 1. Sensat Urban Performance Comparison. Performances of other works except RandLA++ are taken from the SensatUrban paper [16] RandLA++ performance is taken from its paper. [6]
Table 3. DALES Performance Comparison. RandLA results are directly obtained from the author of RandLA. All other results are taken from DALES paper. [35]

<table>
<thead>
<tr>
<th>Method</th>
<th>OA(%)</th>
<th>mIoU(%)</th>
<th>ground</th>
<th>vegetation</th>
<th>car</th>
<th>truck</th>
<th>power lines</th>
<th>poles</th>
<th>fences</th>
<th>building</th>
</tr>
</thead>
<tbody>
<tr>
<td>KPConv [33]</td>
<td>97.8</td>
<td>81.1</td>
<td>97.1</td>
<td>96.6</td>
<td>85.3</td>
<td>41.9</td>
<td>75.0</td>
<td>95.5</td>
<td>63.5</td>
<td>94.1</td>
</tr>
<tr>
<td>RandLA [17]</td>
<td>97.09</td>
<td>79.27</td>
<td>96.99</td>
<td>93.16</td>
<td>82.22</td>
<td>38.6</td>
<td>94.96</td>
<td>58.01</td>
<td>73.56</td>
<td>96.64</td>
</tr>
<tr>
<td>PointNet++ [26]</td>
<td>95.7</td>
<td>68.3</td>
<td>94.1</td>
<td>89.1</td>
<td>75.4</td>
<td>30.3</td>
<td>40.0</td>
<td>79.9</td>
<td>46.2</td>
<td>91.2</td>
</tr>
<tr>
<td>ConvPoint [5]</td>
<td>97.2</td>
<td>67.4</td>
<td>96.9</td>
<td>96.3</td>
<td>75.5</td>
<td>21.7</td>
<td>40.3</td>
<td>86.7</td>
<td>29.6</td>
<td>91.9</td>
</tr>
<tr>
<td>SPGraph [19]</td>
<td>95.5</td>
<td>60.6</td>
<td>94.7</td>
<td>93.4</td>
<td>62.9</td>
<td>18.7</td>
<td>28.5</td>
<td>65.2</td>
<td>33.6</td>
<td>87.9</td>
</tr>
<tr>
<td>PointCNN [30]</td>
<td>97.2</td>
<td>58.4</td>
<td>97.5</td>
<td>95.7</td>
<td>40.6</td>
<td>4.8</td>
<td>57.6</td>
<td>26.7</td>
<td>52.6</td>
<td>91.7</td>
</tr>
<tr>
<td>ShellNet [17]</td>
<td>96.4</td>
<td>57.4</td>
<td>96.0</td>
<td>95.4</td>
<td>32.2</td>
<td>39.6</td>
<td>20.0</td>
<td>27.4</td>
<td>60.0</td>
<td>88.4</td>
</tr>
<tr>
<td>EyeNet(ours)</td>
<td>97.18</td>
<td>79.6</td>
<td>97.15</td>
<td>93.37</td>
<td>83.53</td>
<td>41.3</td>
<td>94.65</td>
<td>57.38</td>
<td>72.54</td>
<td>96.87</td>
</tr>
</tbody>
</table>

Table 4. YUTO Performance Comparison. Results of RandLA and KPConv are taken from internal experiments.

<table>
<thead>
<tr>
<th>Method</th>
<th>mIoU(%)</th>
<th>ground</th>
<th>vegetation</th>
<th>building</th>
<th>water</th>
<th>car</th>
<th>truck</th>
<th>traffic road</th>
<th>sidewalk</th>
<th>parking</th>
</tr>
</thead>
<tbody>
<tr>
<td>RandLA [17]</td>
<td>58.37</td>
<td>80.61</td>
<td>94.44</td>
<td>95.39</td>
<td>3.34</td>
<td>74.59</td>
<td>13.87</td>
<td>78.10</td>
<td>23.43</td>
<td>61.56</td>
</tr>
<tr>
<td>KPConv [33]</td>
<td>56.14</td>
<td>86.94</td>
<td>96.25</td>
<td>94.01</td>
<td>0.00</td>
<td>84.02</td>
<td>0.00</td>
<td>79.93</td>
<td>3.26</td>
<td>60.83</td>
</tr>
<tr>
<td>EyeNet(ours)</td>
<td>63.44</td>
<td>86.26</td>
<td>95.94</td>
<td>96.78</td>
<td>13.61</td>
<td>83.02</td>
<td>14.26</td>
<td>84.65</td>
<td>31.08</td>
<td>65.34</td>
</tr>
</tbody>
</table>

Figure 6. Visualization comparison of SensatUrban between RandLA and EyeNet. From left to right: the input point cloud with RGB colours, semantic segmentation results of RandLA, semantic segmentation results of EyeNet, the ground truth of the input pointcloud.

The results of the performance comparison for Toronto3D are presented in Table 2. The EyeNet model achieved a new state-of-the-art performance for mIoU metric, surpassing all existing methods by a considerable margin. Notably, EyeNet demonstrated the best performance in four out of eight semantic classes. The Toronto3D dataset includes a unique class called road marking, which is challenging to classify without RGB features, and previous approaches have struggled to accurately classify it. However, EyeNet successfully predicted road markings, highlighting its superior performance.

DALES: The Dayton Annotated LiDAR Earth Scan (DALES) [35] dataset is a publicly available dataset of LiDAR point clouds covering an area of approximately 10 square kilometers in Dayton, Ohio, USA and was collected using an Aerial Laser Scanner (ALS). The dataset includes over 505 million points and has an average point density of 25.5 points per square meter. Each point in the dataset is assigned a semantic label from one of 8 classes, including ground, building, vegetation, water, and others. The dataset is divided into 40 sections and split into training, validation, and test sets for semantic segmentation tasks. The evaluation was done by calculating mIoU, OA, and per class IoU for the test split. A voxel size of 0.32 meters was used for grid sampling, and the provided 3D coordinates was used for training and testing the network.

The performance comparison results for DALES are presented in Table 3. The EyeNet model achieved the second-best performance for mIoU metrics. However, since DALES is the most sparse dataset among the tested benchmarks, the benefits of using the human vision based input in EyeNet are minimal.
**YUTO:** The York University Toronto Ontario (YUTO) dataset is a privately owned dataset of LiDAR point clouds covering an area of approximately 9.5 square kilometers in the York University campus. The dataset contains 738 million points collected using an ALS and has an average point density of 78 points per square meter. Each point in the dataset is assigned a semantic label from one of nine classes, including ground, traffic road, sidewalk, water, and others. The dataset is divided into 41 tiles, with 11 tiles used for evaluation. The evaluation was performed by calculating mIoU and per-class IoU for the test split. The performance of two popular semantic segmentation networks, RandLA [17] and KPConv [33], was also evaluated for comparison. A voxel size of 0.20 meters was used for grid sampling, and the provided 3D coordinates and intensity were used for training and testing the network.

The results of the performance comparison for YUTO are presented in Table 4. The EyeNet model outperformed the two tested networks by a significant margin in terms of the mIoU metric. It is noteworthy that EyeNet exhibited the highest performance in six out of the nine semantic categories. The YUTO dataset encompasses classes related to the terrain, including traffic road, ground, sidewalk, parking, and water, which is a distinctive feature of ALS datasets. It poses a challenge to classify these classes without the RGB channel accurately. Nevertheless, EyeNet achieved superior results in recognizing these terrain-related classes.

### 4.3. Ablation Study

An ablation study was conducted to evaluate the effectiveness of the introduced features in the Sensat Urban test set through an online evaluation server.

In order to demonstrate the efficacy of our parallel processing network architecture, we constructed and compared three different types of network structures: Baseline, Sequential, and Parallel. The Baseline structure utilized the pre-existing RandLA-Net architecture. The Sequential structure consisted of two RandLA-Net structures connected sequentially. The Sequential structure was constructed to demonstrate that the performance gain from EyeNet was not solely due to increased parameters. The Parallel structure stacked two RandLA-Net structures in parallel like Fig. 2.

The study compared the performance of four network structures, including the baseline network RandLA, sequentially stacked RandLAs, EyeNet with only Feature Merging Block, and EyeNet. The results, presented in Table 5, indicated that the sequential structure resulted in a 6.7% decrease in mIoU, implying that increasing parameter size alone did not improve performance. In contrast, our EyeNet with only Feature Merging Block yielded improvements of 3.2% and 8.8% in OA and mIoU, respectively, over the baseline. These results suggest that our human vision-inspired approach contributed to performance enhancement. Additionally, the connection block led to further improvements of 0.7% and 0.8% in OA and mIoU, respectively, indicating that the connection between the two streams improved the network’s feature understanding and led to a performance boost.

### 5. Conclusion

Determining an optimal coverage area for the input batch is crucial for effective feature learning in 3D point cloud semantic segmentation, but it is often neglected. To address this issue, we have developed a human vision-based input processing semantic segmentation network architecture, which overcomes the limitations of traditional semantic segmentation networks. We have also introduced a merging block, connection block and parallel stack design for effective feature learning. As a result, our approach has achieved state-of-the-art performance on large scale outdoor benchmark datasets such as SensatUrban and Toronto3D. However, it is possible that the performance of our network may be limited by the baseline network. Therefore, in future work, we plan to investigate the application of our method to different network architectures since our method is model-agnostic.

### 6. Acknowledgment

This research project has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC)’s Collaborative Research and Development Grant (CRD) - 3D Mobility Mapping Artificial Intelligence (3DMMAI) and Teledyne Geospatial Inc. We’d like to thank Alvin Poernomo (Machine Learning Developer), Hamdy Elsayed (Innovation Manager) and Chris Verheggen (SVP R&D).

### References


<table>
<thead>
<tr>
<th>Peri. Input</th>
<th>Structure</th>
<th>CB</th>
<th>OA</th>
<th>mIoU</th>
</tr>
</thead>
<tbody>
<tr>
<td>✗</td>
<td>Baseline</td>
<td>✗</td>
<td>89.8</td>
<td>52.7</td>
</tr>
<tr>
<td>✗</td>
<td>Sequential</td>
<td>✗</td>
<td>89.4</td>
<td>46.0</td>
</tr>
<tr>
<td>✓</td>
<td>Parallel</td>
<td>✗</td>
<td>93.0</td>
<td>61.5</td>
</tr>
<tr>
<td>✓</td>
<td>Parallel</td>
<td>✓</td>
<td>93.7</td>
<td>62.3</td>
</tr>
</tbody>
</table>

Table 5. Ablation results of the peripheral input (Peri. Input), network structure, and connection block (CB).


[36] Lei Wang, Yuchun Huang, Yaolin Hou, Shenman Zhang, and Jie Shan. Graph attention convolution for point cloud semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10296–10305, 2019. 2


[40] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer, 2020. 1, 2, 6