International Journal of Intelligent Robotics and Applications (2024) 8:1085–1101
https://doi.org/10.1007/s41315-024-00376-5

REGULAR PAPER

Comparison of lidar semantic segmentation performance on the structured SemanticKITTI and off-road RELLIS-3D datasets

Mason McVicker[1] · **Lauren Ervin**[1] · **Yongzhi Yang**[1] · **Kenneth G. Ricks**[1]

Received: 16 January 2024 / Accepted: 9 August 2024 / Published online: 20 August 2024

Abstract
Existing lidar-based semantic segmentation algorithms and datasets focus on autonomous vehicles operating in urban environments. This has greatly improved the safety and reliability of these autonomous vehicles in predictable scenery. A new dataset provides lidar data focusing on off-road environments as seen by autonomous ground vehicles, ushering in a new era of off-road exploration capabilities. To the best of our knowledge, no new algorithms have been developed specifically for this unstructured environment. To gain an understanding of how existing algorithms perform in an off-road environment, we assess the baseline performance of four algorithms, KPConv, SalsaNext, Cylinder3D, and SphereFormer, on a commonly used on-road dataset, SemanticKITTI. We then compare the results with an off-road dataset, RELLIS-3D. We discuss the degradation of each algorithm on the off-road dataset and investigate potential causes such as class imbalance, inconsistencies in the labeled data, and the inherent difficulty of segmenting off-road environments. We present the strengths and weaknesses of each algorithm's segmentation abilities and provide a comparison of the runtime of each algorithm for real-time capabilities. This is crucial for identifying what network architecture features are potentially the most beneficial for unstructured scenes. A robust, open-source software implementation via docker containers and bash scripts provides simple, repeatable execution of all algorithm training and evaluations. All code is publicly available at https://github.com/UA-Lidar-Segmentation-Research.

Keywords Semantic segmentation · Lidar · Off-road · Dataset · SemanticKITTI · RELLIS-3D

1 Introduction

Perception is a fundamental part of life. Vision allows animals and insects to navigate, traverse unknown areas, and avoid obstacles. In many ways, robots are the same. Advances in perception technology and algorithms have enabled improvements in multiple fields. They have improved the accuracy and safety of self-driving cars on the roads and autonomous ground vehicles (AGVs) off-road. Off-road navigation has distinct scene awareness requirements, such as elevation mapping and negative obstacle detection to distinguish between traversable and un-traversable regions. In the automotive field, these advances have provided greater safety for drivers and pedestrians and have enabled the elderly and disabled greater independence. AGVs help people carry out time-sensitive and physically demanding search and rescue operations more effectively and with less risk to human life, reduce the risks associated with heat and sun exposure by automating lawn mowing, and reduce casualties due to undetected and unexploded ordinances through autonomous humanitarian demining operations, among other benefits (Wang 2021; Wigness et al. 2019; Kopacek 2004; Kushwaha et al. 2016; Nagatani et al. 2013; Murphy 2014).

The sensors used are central to these applications. Camera, radar, ultrasonic, and lidar sensors are commonly used, and each has distinct advantages and disadvantages. Lidar works by sending a series of pulses of light at a specific frequency and observing the reflection off of the target.

✉ Mason McVicker
mmcvicker@crimson.ua.edu

Lauren Ervin
lefaris@crimson.ua.edu

Yongzhi Yang
yyang108@crimson.ua.edu

Kenneth G. Ricks
kricks@eng.ua.edu

[1] Department of Electrical and Computer Engineering, University of Alabama, Tuscaloosa, AL 35487, USA

The time it takes the light to return and the amount of light reflected are sensed, and a distance is calculated. This distance is used to generate a point in 3D, relative to the sensor. Several emitters and detectors are stacked vertically and spun in a circle to provide a high density three-dimensional scan of the area surrounding the sensor, commonly known as a point cloud. Like radar and ultrasonic sensors, lidar is an active sensor that emits the light it needs to observe the world. Because of this, it operates equally well under all lighting conditions. Lidar also has long detection ranges and high measurement accuracy (Wang 2021).

Changes in viewpoint dramatically affect the appearance of an object (Li et al. 2020). For example, when passing a tanker truck, lidar scans of the front and rear are very different (Eastepp et al. 2022). The front is a traditional truck cab, while the rear is round and cylindrical. This variation makes hand-crafting a set of rules or features to interpret different lidar scans of even a single object difficult. Semantic segmentation, along with object detection, classification, and localization, forms the foundation for autonomous vehicles (Li et al. 2020). Detection and classification can be extracted from the output of a semantic segmentation algorithm, and localization is an inherent property of lidar data. To help progress the state of autonomous vehicles and AGVs, we will focus this work on evaluating semantic segmentation algorithms.

In order to learn rules and features, machine learning algorithms require large amounts of labeled data. Labeled data includes raw point clouds provided by the sensor and a set of annotations that specify to which class each of the points in the point cloud belongs. Because the algorithm learns directly from the labeled data, it is important that the labels are accurate and consistent. Large-scale datasets such as SemanticKITTI (Behley et al. 2019), the Waymo Open Dataset (Sun et al. 2020), nuScenes (Caesar et al. 2020) and others (Geyer et al. 2020; Roynard et al. 2018; Fong et al. 2021; Huang et al. 2020) contain manually labeled data collected by driving around cities with sensors mounted to the vehicle. They provide a solution to the requirement for high-quality labeled data with publicly available datasets for researchers to download and use. Since this data is available, various research groups have created algorithms designed for them (Aksoy et al. 2019; Yan et al. 2022; Hu et al. 2020; Zhu et al. 2021; Lai et al. 2023; Tang et al. 2020; Liu et al. 2019; Kong et al. 2023), furthering research progress.

All of the datasets and algorithms mentioned above are designed for on-road driving in a city. Because of the inherent structure of cities, such as flat roads, vertical buildings, and other man-made objects, we will call this a "structured" environment, and datasets containing data collected in this environment will be called "structured" datasets. In contrast to the structure of the man-made world is the lack of structure of the natural world, which we will call an "unstructured" environment and the datasets collected here "unstructured" datasets because of the lack of regularity in the structure of the environment. In unstructured environments, there is generally no clearly traversable path for AGVs to drive on, instead relying on sensing to determine where to drive. Only recently have datasets been published for these unstructured environments. RUGD (Wigness et al. 2019) and RELLIS-3D (Jiang et al. 2022) are two new unstructured datasets. RUGD includes only camera images and is therefore not directly applicable to this work, but RELLIS-3D contains both camera images and lidar point clouds. To our knowledge, no new machine learning algorithms have been created for RELLIS-3D, and only two algorithms have been evaluated on the dataset when initially published. Again, these two algorithms were originally developed for structured data not commonly found in the unstructured world present in the RELLIS-3D dataset.

Comparing the performance of different algorithms is important for researchers and professionals to choose the best option for their application. Several methods have been created in an effort to do this. A well-known tool is the Papers With Code (The latest in Machine Learning. https://paperswithcode.com/). The website, created by Meta AI, contains more than 100,000 papers with published results spanning more than 8500 datasets, including all of the datasets mentioned in the previous paragraphs. Some dataset publishers also host leaderboards for the performance of algorithms on their dataset. These include the Waymo Open Dataset Leaderboard (Sun et al. 2020) and the SemanticKITTI CodaLab competition page (Papers With Code—The latest in Machine Learning. https://paperswithcode.com/). These leaderboards provide useful information, but have some flaws. The first flaw is that the hardware and amount of training data is not standardized. Different hardware can contribute significantly to the performance of the algorithm (Li et al. 2017). Additionally, some algorithms claim to use additional training data, while others do not. This benefits algorithms that use extra training data, as more data generally improves performance. Furthermore, in the case of the Papers With Code page, the results are self-reported. Although the results are supervised by a volunteer group, there are likely to be inconsistencies as the time required to verify and validate the results of each paper is prohibitive.

Contributions: In order to eliminate the quantity of training data as a variable and to gain an understanding of how algorithms designed for structured environments work in the unstructured world, we compare the performance of several state-of-the-art lidar semantic segmentation algorithms on SemanticKITTI, a structured dataset, and RELLIS-3D, an unstructured dataset. Training and evaluation are performed on identical hardware across networks and datasets, eliminating hardware as a potential performance parameter. Both datasets use the same labeling scheme, providing

consistency in data reading methods and minimizing the code changes needed to run algorithms on each dataset. We compare four different algorithms; KPConv (Thomas et al. 2019), SalsaNext (Cortinhal et al. 2020), Cylinder3D (Zhu et al. 2021), and SphereFormer (Lai et al. 2023).

2 Related works

2.1 Datasets

Multimodal datasets targeting various objects often use some combination of camera, lidar, radar, Global Positioning System (GPS), and Inertial Measurement Unit (IMU) sensors to collect correlated data. Even within similar raw data, there are different labeling schemes including bounding boxes, semantic labels, and vehicle pose. Additionally, the focus of these datasets can be classified as on-road, off-road, indoor, or objects. These datasets are frequently used for semantic segmentation, scene completion, pose tracking, object tracking, and object classification. We focus on on- and off-road autonomous driving lidar datasets with the goal of performing semantic segmentation.

The Waymo Open Dataset (Sun et al. 2020), SELMA (Testolina et al. 2023), SemanticKITTI (Behley et al. 2019), and others (Geyer et al. 2020; Caesar et al. 2020; Huang et al. 2020) contain data from vehicles driving in cities, all with lidar data from a roof mounted spinning lidar sensor, and most contain other sensor and positioning data. RUGD (Wigness et al. 2019) and RELLIS-3D (Jiang et al. 2022) are recorded by small AGVs in outdoor environments. Both contain camera images, and RELLIS-3D also contains lidar data. As this work uses the SemanticKITTI and RELLIS-3D datasets extensively, we will discuss them in greater detail.

2.1.1 SemanticKITTI

The SemanticKITTI dataset is based on the KITTI dataset (Geiger et al. 2012). KITTI used sensors mounted on a vehicle to record 22 sequences of data including images and lidar scans. Each sequence is a separate drive around the city of Karlsruhe, Germany and contains a list of the timestamps of the camera and lidar scans with the 3D pose of the scan. This dataset can be used for stereo, optical flow, visual odometry, simultaneous localization and mapping (SLAM), and 3D object detection. Approximately seven years later, a different research team compiled the SemanticKITTI dataset to provide semantic labels for the lidar scans in all 22 sequences. In total, there are over 40,000 scans with more than 4.5 billion points that are all labeled with class annotations. There are 19 training classes represented in the dataset including road, sidewalk, parking, other-ground, building, car, truck, bicycle, motorcycle, other-vehicle, vegetation, trunk, terrain, person, bicyclist, motorcyclist, fence, pole, and traffic sign. There is also other-structure and other-object classes that are omitted for evaluation. Of the 22 sequences of scans, 10 are dedicated to the training set, totaling 23,201 scans, with one sequence for validation and the remaining 11 for testing, totaling 20,351 scans (Fig. 1).

2.1.2 RELLIS-3D

With inspiration from SemanticKITTI and RUGD, the RELLIS-3D dataset was developed. This dataset contains camera images, lidar scans, and robot pose. It was collected by driving an AGV around the Rellis campus of Texas A &M University and contains 13,556 lidar scans and 6235 camera images, divided across 5 sequences. Twenty classes were annotated, including sky, grass, tree, bush, concrete, mud, person, puddle, rubble, barrier, log, fence, vehicle, object, pole, water, asphalt, building, and dirt. Approximately 80% of the lidar points are contained in the classes of grass, tree, and bushes, showing a large imbalance in the dataset. The training set for this dataset contains 7800 scans with subsequences from four of the sequences. There are 2413 scans in the validation set, comprising subsequences from two of the sequences. Finally, there are 3343 scans in the test set containing subsequences from three of the sequences. They state the splits have been done this way to create a large training set with a representative testing and validation set (Fig. 2).

2.2 Algorithms/networks

As there are no specific networks designed for semantic segmentation of lidar point clouds in unstructured environments, algorithms and literature related to general point cloud segmentation and structured environment segmentation are investigated. The algorithms were selected for three main reasons:

1. They all have publicly available pytorch implementations;

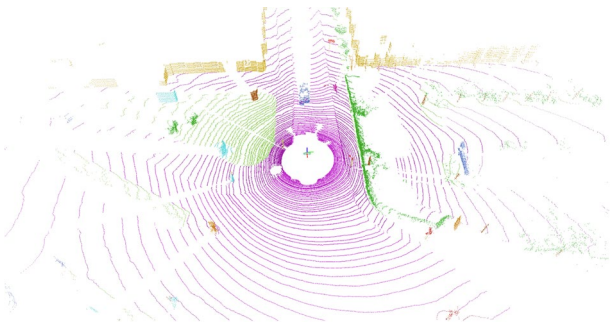

Fig. 1 A labeled scene from the SemanticKITTI dataset (colors indicate different classes of objects)

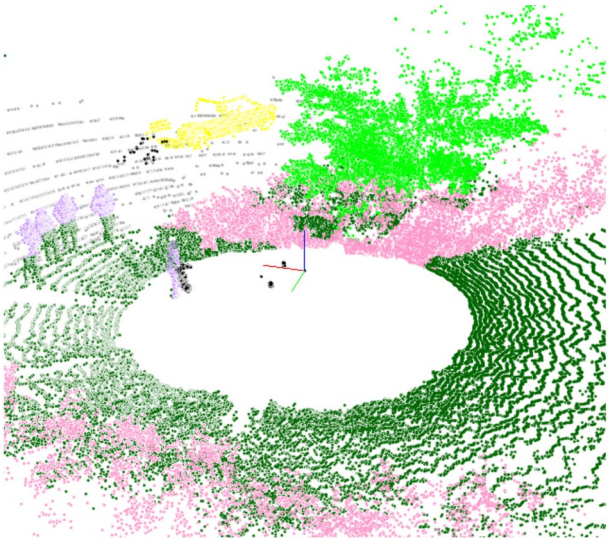

Fig. 2 A labeled scene from the RELLIS-3D dataset

2. They had the highest mIOU scores on the SemanticKITTI dataset at the beginning of this project in August 2023;
3. Each algorithm had a unique architecture that distinguished it from the others.

2.2.1 KPConv

KPConv (Thomas et al. 2019) expands the popular convolutional neural network (CNN) to 3D point clouds by using the kernel directly in 3D space rather than first converting from a point cloud to a range image like (Kong et al. 2023; Aksoy et al. 2019; Cortinhal et al. 2020). Most CNN implementations rely on grids of data, such as the pixel structure of a camera image. A convolutional kernel operates on this grid, collecting information from the image through convolution, which is then used in segmentation or classification. KPConv introduces the Kernel Point Convolution, called KPConv, which is a new point convolution operator where kernel points are defined in 3 dimensions as points. Points in the target point cloud are then correlated to the points in the convolution.

This novel approach was the model with the highest ranking on the SemanticKITTI leaderboard on Papers With Code for approximately a year, achieving a mean intersection-over-union (mIOU) score of 58.8 (Behley et al. 2019), only supplanted by the next network on our list, SalsaNext. KPConv was also reported in the RELLIS-3D paper with an mIOU of 19.97 Jiang et al. (2022).

2.2.2 SalsaNext

SalsaNext (Cortinhal et al. 2020) is the next iteration of SalsaNet (Aksoy et al. 2019). Both of these networks use a projection-based method, where the points are projected back onto a cylinder around the lidar and turned into a 5D range-view image that is processed like a camera image. SalsaNext uses an encoder-decoder architecture where the range-view image has several levels of convolution with various kernel sizes, dilation rates, and batch normalization until it reaches a minimal representation of the data. The convolution is then reversed to revert to the original image size. Each point will also have 20 dimensions, equal to the number of classes, with the most activated of these dimensions being the output class.

SalsaNext differs from SalsaNet in two major ways, using a pixel-shuffle layer instead of a traditional deconvolution, and using an uncertainty estimation approach to account for the noise inherent to lidar sensors. The pixel-shuffle layer takes the extra dimensions in the channels and redistributes them to the height and width spatial dimensions. This has the effect of allowing more of the channel representation at the center layer of the architecture to contain more information about the spatial relationships in the data while still preserving the compression of the encoder-decoder architecture. The uncertainty estimation replaces the output predictions with probability distributions based on propagating sensor noise through the network. They also propagate weight uncertainty through a Bayesian Neural Network (BNN) to capture irreducible uncertainty in the data.

Using this uncertainty measurement and the pixel-shuffle layer allowed SalsaNext to achieve state-of-the-art performance on the SemanticKITTI dataset, achieving an mIOU of 59.5 and an mIOU score on the RELLIS-3D dataset of 43.07 (Jiang et al. 2022). KPConv and SalsaNext were the two networks used for the lidar data in the RELLIS-3D paper.

2.2.3 Cylinder3D

Cylinder3D (Zhu et al. 2021), like KPConv, does not rely on projection of the 3D point cloud into a 2D image the way that SalsaNet, SalsaNext, and others do. It also avoids partitioning the world into square voxels, instead partitioning into cylindrical coordinates, and utilizes asymmetrical residual blocks. Cylindrical coordinate partitioning is performed by first converting the points into cylindrical coordinates, namely (ρ, θ, z) rather than the traditional (x, y, z). In parallel, the point cloud is fed through a series of multi-layer perceptrons (MLPs) to gather point-wise features. These two steps are combined to create a set of cylindrical features through cylindrical partitioning, represented in Fig. 3, which are fed through an encoder-decoder network with asymmetrical residual blocks. The asymmetrical residual blocks have two

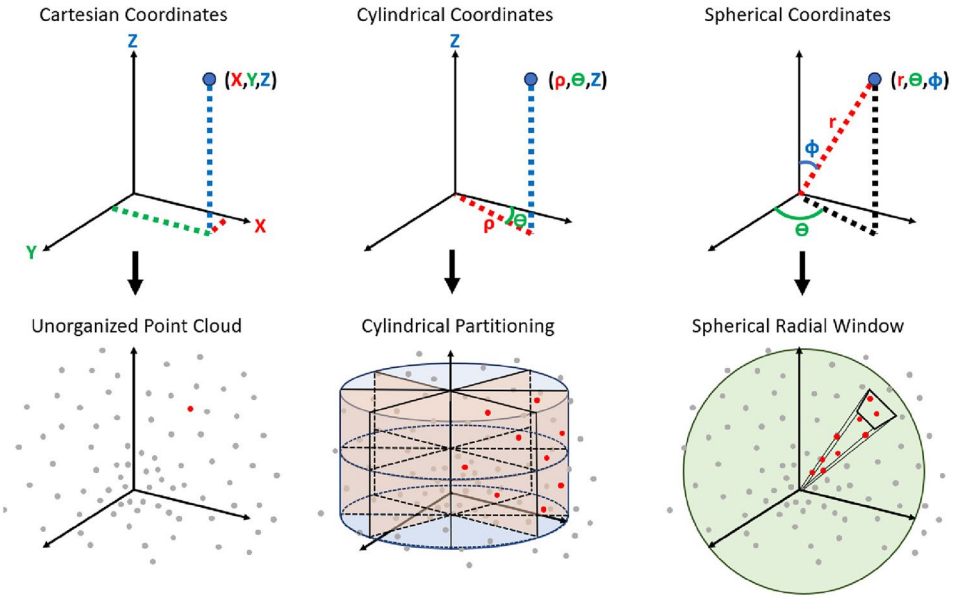

Fig. 3 Cylindrical partitioning and spherical radial window

branches that each do a convolution along the ρ and θ axes, with θ following ρ on one side and ρ following θ on the other. The two branches are concatenated before the downsampling convolution in the encoder and after the deconvolution is concatenated with the features from the other side of the decoder. The asymmetry enhances the robustness of the algorithm.

Using these innovations, Cylinder3D was able to achieve state-of-the-art performance with a reported mIOU of 67.8 on SemanticKITTI. As the network was published after the RELLIS-3D dataset and to the best of our knowledge no one else has run this network on the RELLIS-3D data, there is no available data for comparison.

2.2.4 SphereFormer

Recently there has been an explosion of new networks (Kong et al. 2023; Guo et al. 2021) using the Transformer (Vaswani et al. 2023) architecture. SphereFormer (Lai et al. 2023) applies the popular transformer architecture to 3D point clouds. To do this, they use the U-Net (Ronneberger et al. 2015) backbone and SparseConv (Graham Maaten 2017; Graham et al. 2017) as a baseline model, similar to Cylinder3D. They added a radial window partition and exponential splitting of the r dimension, the distance from the sensor, denoted ρ in the Cylinder3D section. This divides the 3D space into angular segmentations in θ and ϕ. These segmentations are then segmented by range in an exponential mapping. This makes the bins closer to the sensor smaller, and the bins farther away bigger, shown in Fig. 3. Because lidar sensors have a higher point density close to the sensor, this helps to capture a similar number of points in each bin.

The SphereFormer model was able to achieve a state-of-the-art mIOU of 74.8 on the SemanticKITTI dataset. To our knowledge, the model has not yet been evaluated on the RELLIS-3D dataset.

3 Methodology

3.1 Hardware

Training was performed on a computer executing on Ubuntu 20.04 with an Intel i9-9960X CPU, 64 GB of DDR4 RAM, and two NVIDIA Quadro RTX 8000 GPUs with 48 GB of VRAM per card. These GPUs are bridged together utilizing NVIDIA's NVLINK Bridge, enabling the GPUs to share the VRAM present on each card. As a result, the training process had 96 GB of VRAM available. To minimize thermal throttling, which could negatively affect training performance, case fans and a CPU cooler were used. The training hardware is better than some of the original hardware used for training the four considered networks. Rather than trying to match all the original training hardware from each of the published networks, our training hardware sets a baseline on which all four networks can be evaluated. Thus, hardware is not considered to be a parameter for comparison in this work.

3.2 Software infrastructure

In an effort to simplify the environment setup, data preparation, and training process, we have invested into a robust software environment. This includes docker containers, data format preparation scripts, and a unified training process for

all four networks and both datasets. This was accomplished primarily with Docker and bash scripts. Details for the individual parts of the software infrastructure are included in this section.

3.2.1 Docker containers

To simplify the process of installing different sets of dependencies, to streamline the training process, and to manage repeatable environment setups, we created a docker container for each network. Each container is generated from a dockerfile which contains a list of commands used to set up the software environment, including installing dependencies and running setup scripts. The dockerfile also sets the working directory to the location with the network code.

To build the docker container with the desired settings, a complicated "docker build" command must be executed with the correct parameters. To simplify this process, we wrote a bash script that builds the container with the appropriate options. To use the newly created software environment, a terminal is opened in the docker container using a "docker run" command. Like the "docker build" command, this has a complicated set of parameters to ensure that the correct options have been set. We wrote another bash script which handles the rest of the setup and executes the docker run command. The general environment infrastructure is shown below in Fig. 4.

With the use of both scripts, the environment setup is simplified to a small set of commands. This eliminates the time-consuming tasks of setting up software environments and installing compatible libraries and software packages. The setup is also portable to multiple different PCs, so all training and evaluations are repeatable on similar hardware.

Research (Felter et al. 2015) has shown that there is "negligible overhead for CPU and memory performance" when using docker. Additionally, the NVIDIA libraries used within the docker environment have direct access to the GPU hardware. As such, we are confident that using docker containers to host the software environment has not caused a degradation of performance in the model regarding accuracy or inference runtime. Any possible overhead would remain consistent across all four models and both datasets.

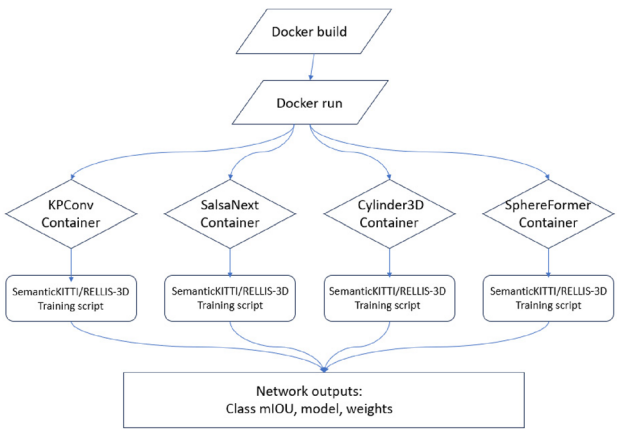

Fig. 4 Environment architecture

3.2.2 Train validation test split

SemanticKITTI and RELLIS-3D have different numbers of sequences and a different distribution of scans within those sequences. In an effort to compare the two datasets as directly as possible, we split the data in a similar manner for both. RELLIS-3D contains fewer sequences and fewer scans than SemanticKITTI, so we will use its train, validation, and test splits as a baseline. We will then match the number of scans for the SemanticKITTI split by intentionally capping the total data used to 7800 scans. We will also match the distribution of scans within sequences to the best of our ability.

The training, validation, and testing split distribution of sequences and scans within each sequence for the RELLIS-3D dataset is shown in Table 1. The first column contains an enumeration of the five sequences in the dataset. The "Training" column contains the start and end scans of each sequence that were added to the training set. For example, from sequence 0, scan number 307, 308, ..., 1705, inclusive, were added to the training set. The "Validation" and "Testing" columns show similar data. At the bottom of the table is the total number of scans for each of the three sets.

We chose the same split for SemanticKITTI, with SemanticKITTI sequences matched to the RELLIS-3D sequence with the most similar number of scans, as shown in the right column of Table 1. We were unable to match the number exactly on RELLIS-3D sequence 4, so we augmented SemanticKITTI sequence 9 with 468 scans from sequence 3. This split was done without prior knowledge of the composition of the contents of any of the sequences to ensure that the SemanticKITTI dataset did not receive more favorable conditions than RELLIS-3D.

3.2.3 Algorithm preparation

Each algorithm required minor changes from the published implementation for compatibility with RELLIS-3D. We add a "rellis.yaml" file where needed to describe the dataset. Like the "semantic-kitti.yaml" file, this describes the class structure of the dataset and the train, validation, and test split. Additional changes specific to each network are outlined below.

SalsaNext In SalsaNext, there are calculations that use the depth of a point. The SemanticKITTI dataset uses a Velodyne sensor that omits points with no return, while the RELLIS-3D dataset uses an Ouster sensor that sets all fields

Table 1 Sequence distribution for RELLIS-3D dataset

RELLIS-3D		Training		Validation	Testing	SemanticKITTI
0	Start	307		1706	0	0
	End	1705		2849	306	
1	Start	–		1047	0	5
	End	–		2318	1046	
2	Start	0		–	2158	2
	End	2157		–	4146	
3	Start	0		–	–	8
	End	2183				
4	Start	0	0	–	–	9
	End	2058	1591			
	Start	–	0	–	–	3
	End	–	467			
Total		7800		2413	3343	

of points with no return to 0. Due to this, the SalsaNext algorithm encounters a divide-by-zero error when training on the RELLIS-3D dataset. To counteract this issue, we have followed the example of the authors of the RELLIS-3D paper and introduced an additional operation on points that have 0 depth, instead setting it to 1e−4 shown below in Algorithm 1. Additionally, the field of view parameter was changed to match the Ouster.

Algorithm 1 Spherical projection

Require: Raw point cloud
Ensure: Performs spherical projection on point cloud
1: Set FoV
2: Retrieve X,Y,Z points
3: Get depth of all points
4: **if** depth == 0 **then**
5: depth = 1e-4
6: **end if**
7: Yaw = -arctan2(Y,X)
8: Pitch = arcsin($\frac{Z}{depth}$)
9: Get projections in image coordinates
10: Scale to image size using angular resolution

Cylinder3D The cylindrical partition used in Cylinder3D has bounds set by a parameter in the config files. For SemanticKITTI, these were set from −4 to +2 m in the Z direction in order to bound 99%+ of the points in the scan (Zhu et al. 2021). When switching to the RELLIS-3D dataset, we expanded the Z bound from −4 to +4 m in order to capture the same percentage of points collected from the Ouster.

SphereFormer The training config files for SphereFormer have been changed to use two GPUs instead of four. The SphereFormer paper (Lai et al. 2023) presents training using four GeForce GTX 3090 GPUs, for a total of 96 GB. Although our hardware is not identical, we have matched the total amount of VRAM available with our two NVIDIA Quadro RTX 8000 GPUs.

3.3 Training and evaluation process

The first step of the training process is to build and run the docker container. Next, one of the two utility training scripts, for SemanticKITTI or RELLIS-3D, is used to train the network. The output of the training process is displayed on the terminal and saved to a file. We save the model including weights for the best validation mIOU score for evaluation. The mIOU is defined as

$$IOU = \frac{TP}{TP + FP + FN},$$

$$mIOU = \frac{1}{C}\sum_{i=1}^{C} IOU(i)$$

where TP is true positive, FP is false positive, FN is false negative, and C is the number of classes. The maximum mIOU score is 1, representing a perfect prediction, but we present mIOU scores scaled by percent. This comparison of ground truth and prediction is made to estimate accuracy and evaluate network performance. The inference is assessed by a separate evaluation script for each network from within the respective docker container. Inference time is defined as

$$t_{\text{inference}} = \frac{1}{N}\sum_{i=1}^{N} t_e(i) - t_s(i)$$

where t_s refers to the time right before the prediction process starts, t_e is recorded directly after the execution, $t_{\text{inference}}$ is the average inference time, and N is the number of predictions performed. After inference is performed on each point cloud, the script saves the predicted labels. A separate program from the semantic-kitti-api is used to evaluate the mIOU scores. Using an external method of determining the mIOU score ensures that there is no variance in the method of computing score and that the score is computed over all points in

the scan. We use the "evaluate_semantics.py"python script running on the laptop computer with no code changes. The labeled data and predicted labels are provided to the script with the dataset configuration file. This script computes an overall mIOU score along with the mIOU scores for each class. The overall score for each algorithm on each dataset, as well as the individual class performance is provided in Sect. 4. The evaluation scrips also save the system time before and after the inference call and then compute the average inference time for all of the test set. This is also reported in Sect. 4.

4 Results

4.1 SemanticKITTI quantitative performance

Given the significantly smaller amount of training data used for this experiment than the published results for each of the four networks tested, we expect worse results for the SemanticKITTI dataset. Table 2 shows this to be true for all networks; KPConv is 9.4% mIOU worse than the published value, the closest of the four, and SphereFormer is 16.7 % mIOU worse, the farthest from the published value. SalsaNext and Cylinder3D fall in the middle at 8.6% and 8.7% worse, respectively.

We believe KPConv is the closest to the published performance in part because it was not trained with extra training data, as shown on the Papers With Code competition entry. SalsaNext was also not trained with extra training data, and is the next closest to the published results. This shows the importance of a large training set for the overall performance of a machine learning model.

Neither the SphereFormer Papers With Code competition entry nor published paper mention what kind or how much extra training data was used. We suspect that the exponential splitting of the range dimension and the self attention mechanisms in SphereFormer may require more data to work as effectively as possible, while the deformable convolution in KPConv does not require as much data, as it deforms to the data at hand in each scan. Despite this, SphereFormer still performed 8.7% better than KPConv with the same data.

Table 2 Our SemanticKITTI mIOU results compared with published results

Network	Published	Ours	Change (%)
KPConv	58.8	49.4	− 9.4
SalsaNext	59.5	50.9	− 8.6
Cylinder3D	67.8	59.1	− 8.7
SphereFormer	74.8	58.1	− 16.7

Cylinder3D was the best performer on our test with an mIOU of 59.1. We suspect that the asymmetrical residual block from Cylinder3D requires less data to train effectively than the self-attention mechanism used in SphereFormer. From the published results, we see that with more data, the self-attention performs better, but from our results, we see that with less data, the asymmetrical residual block performs better.

The class results shown in Table 3 give more insight into the above discussion. In each column, the algorithm that performs best is shown in **bold**, and the worst is *italicized*. SphereFormer performed the best in 9 out of 19 classes, but only Person and Motorcyclist were significantly better than Cylinder3D, which was the best in five of the remaining classes. Cylinder3D did significantly better at Other-Vehicle and Bicycle, and slightly better at the other classes. Overall, Cylinder3D and SphereFormer are comparable. KPConv did the best at five other classes. SalsaNext performed poorly across the board, proving the worst in 13 classes. This may be partially compensated for by the run-time speed of SalsaNext for some applications.

4.2 RELLIS-3D quantitative performance

Table 4 shows the general performance of each network on RELLIS-3D and Table 5 shows the performance of the network on each class in the RELLIS-3D dataset. In each column, the algorithm that performs best is shown in

Table 3 SemanticKITTI class mIOU results where Cyl3D = Cylinder3D and SphFor = SphereFormer

Class	KPConv	SalsaNext	Cyl3D	SphFor
Car	94.69	*88.5*	**95.79**	94.56
Bicycle	*19.8*	25.7	**37.02**	27.49
Motorcycle	*27.37*	29.8	**55.42**	54.98
Truck	**4.23**	*0*	0.55	0.04
Other-vehicle	23.86	23.8	**41.24**	*12.05*
Person	*25.78*	39.7	47.9	**58.43**
Bicyclist	*56.87*	76.4	84.19	**88.85**
Motorcyclist	*0*	*0*	1.6	**10.46**
Road	94.78	*94.6*	96.53	**96.86**
Parking	*15.23*	61.0	69.73	**71.57**
Sidewalk	85.83	*84.7*	87.66	**88.12**
Other-ground	**1.4**	*0*	0.38	0.5
Building	**92.59**	*83.5*	90.73	90.62
Fence	68.39	*61.8*	69.15	**70.21**
Vegetation	86.22	*80.8*	86.32	**86.66**
Trunk	65.68	*58.0*	**73.92**	72.44
Terrain	**68.71**	*51.7*	60.69	60.03
Pole	**60.83**	*48.6*	59.28	54.63
Traffic-sign	*46.71*	59.3	65.49	**66.13**

Table 4 Our RELLIS-3D mIOU results compared with published results

Network	Published	Ours	Change (%)
KPConv	19.97	25.21	+5.2
SalsaNext	43.07	34.25	− 8.82
Cylinder3D	–	46.07	–
SphereFormer	–	42.19	–

Table 5 RELLIS-3D class mIOU results where Cyl3D = cylinder3D and SphFor = SphereFormer

Class	KPConv	SalsaNext	Cyl3D	SphFor
Grass	*60.4*	64.16	**66.58**	66.48
Tree	73.8	*67.52*	77.72	**79.48**
Pole	51.8	*44.27*	**70.57**	56.37
Water	0	0	0	0
Vehicle	*2.9*	17.59	**60.16**	22.37
Log	*0*	0.94	*0*	**8.1**
Person	*69.9*	82.26	**86.67**	83.72
Fence	*0.9*	1.97	8.88	**9.86**
Bush	69.4	*68.4*	**73.06**	71.71
Concrete	*8.1*	53.26	80.67	**84.22**
Barrier	*13.8*	45.12	**82.93**	76.8
Puddle	*1.9*	24.88	**23.54**	9.24
Mud	*0.1*	8.97	13.97	**17.48**
Rubble	*0*	0.13	0.23	**4.81**

bold, and the worst is *italicized*. Notable is the mIOU for the Water class. Upon visual inspection, we were unable to find any Water points in the test set. The authors of the RELLIS-3D paper omitted other classes from the lidar portion of the dataset that contained very few points, but left the Water class, as it is not traversable by AGVs. In order for this class to perform the intended function of segmenting un-traversable areas, the presence of Water points would need to be increased.

Certain classes such as Log, Fence, Puddle, Mud, and Rubble performed poorly across all of the networks as shown in Table 5. These have some of the lowest numbers of points in the dataset. Barrier, Vehicle, and Pole also have low numbers of points; however, these three classes have more distinctive shapes than the others and therefore performed better. All barriers in the dataset are angular traffic barriers with flat faces, which contrasts sharply with the organic shapes of the natural classes. Similarly, poles are vertical and have a distinct cylindrical shape with few surrounding points.

4.3 RELLIS-3D qualitative performance

In this section, we present several algorithm outputs for visual comparison.

Each picture contains five scans in the same orientation. From left to right and top to bottom, there are: labeled training data, KPConv, SalsaNext, Cylinder3D, and SphereFormer outputs. Figure 5 shows a scan in which Cylinder3D was able to successfully segment the vehicle (yellow) when none of the other algorithms were. KPConv also segmented the person incorrectly, annotating them as Bush. In the labeled scan, the people on the left are labeled half Person and half Grass, and the person near the center has several Void points.

In Fig. 6, the grass field is predicted correctly by all algorithms except KPConv, which added Bush and Rubble. All except KPConv also predicted Bush, Barrier, Person, Tree, and Concrete well. KPConv classified the person as Bush, the Barrier as half Bush and half Barrier, and the Concrete as a mixture of Grass, Fence, and Vehicle.

The qualitative analysis of these algorithms confirms the trend seen in Table 5. The classes we see that performed well numerically also performed well visually. The Vehicle points seen in Fig. 5 are a prime example of this. All algorithms except Cylinder3D performed poorly on the Vehicle class, and all outputs except Cylinder3D showed no Vehicle points on the vehicle. Cylinder3D, however, performed well numerically and is confirmed as all of the points are labeled correctly in the scan.

4.4 Comparison of SemanticKITTI and RELLIS-3D performance

All four networks performed worse on the RELLIS-3D dataset than they did on SemanticKITTI, shown in Table 6. The difference between the two is between 13.0% and 24.2% mIOU. We expected this might happen due to the difficulties segmenting unstructured data, class imbalance, and issues present in the RELLIS-3D dataset, discussed below.

Table 6 Comparison of our SemanticKITTI and RELLIS-3D mIOU results

Network	SemanticKITTI	RELLIS-3D	Change (%)
KPConv	49.4	25.2	−24.2
SalsaNext	50.9	34.2	−16.7
Cylinder3D	59.1	46.1	−13.0
SphereFormer	58.1	42.2	−15.9

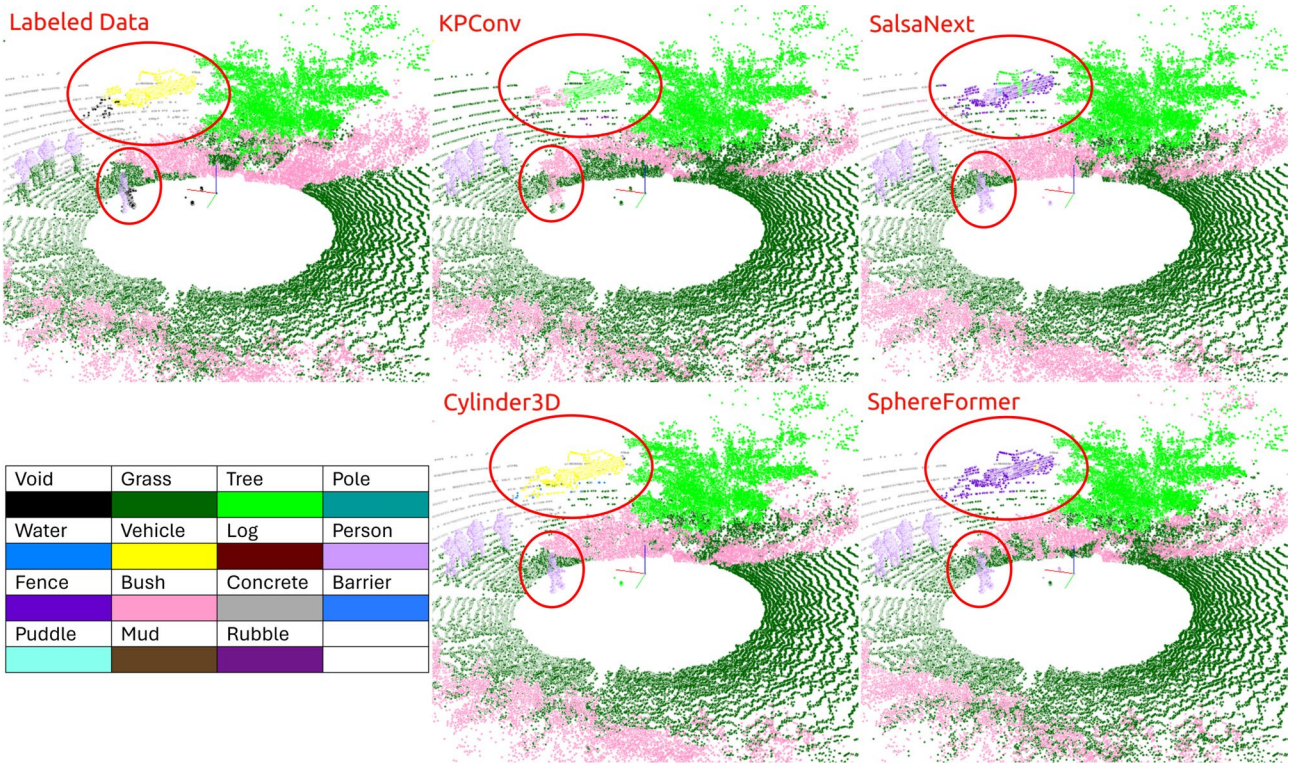

Fig. 5 Sequence 1 scan 130

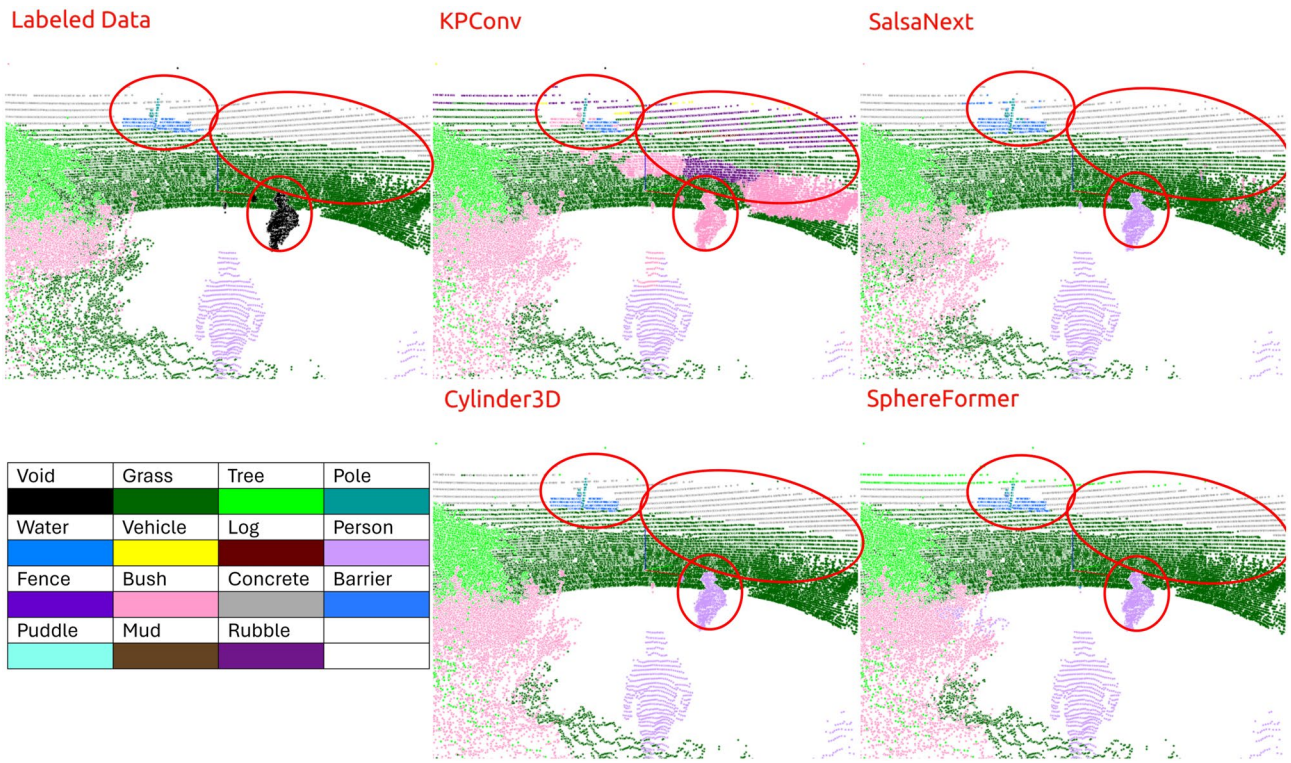

Fig. 6 Sequence 2 scan 3108

4.4.1 Difficulties in segmenting unstructured data

Structured Environments generally contain lots of sharp angles and distinct boundaries. It is easy, for example, to tell where pavement ends and a building begins. It it much harder to determine the boundaries between classes in an unstructured environment. Some examples of classes that are difficult for both the human labeler and the trained network to distinguish include, but are not limited to, the following. The difference between what should be considered Tree and what should be considered Bush can be unclear. A large bush could be considered a small tree, and vice-versa. A grassy field with small bushes, spaced such that the AGV cannot pass but the lidar sensor can still see the underlying grass, also presents a difficult scenario. The human labeler is forced to decide what should be classified as Grass and what should be classified as Bush. In this scenario, we were unable to determine the differences between Grass and Bush. In the RELLIS-3D dataset, we see scans where the ground under and surrounding the Bush was labeled Grass, and other scans where it was labeled Bush. As human labelers were unable to consistently determine the differences and boundaries between the classes, we expect the trained algorithm to have similar difficulties.

We believe that the inherent difficulty in segmenting unstructured data contributes to the poor performance of each network on RELLIS-3D when compared to SemanticKITTI. However, we are unable to quantify exactly what effect the inherent difficulty of segmenting unstructured environments has on the overall performance of each network.

4.4.2 Class imbalance

A known issue with the RELLIS-3D dataset is class imbalance. There is a four-order-of-magnitude difference in the number of points belonging to the most and least represented class. There are ten times more points in each of the Grass, Tree, and Bush classes than any other, and ten times fewer points in the Pole class than any other. Interestingly, the pole class was one of the better performing classes in the dataset. We believe that this is due to the geometry and positioning of the Pole class in the dataset. Poles are generally straight and are not close to any other objects.

SemanticKITTI also has a large disparity of four orders of magnitude between the most represented and least represented classes, with four classes with similarly high numbers of points and three classes with very low numbers of points. This is similar to the distribution of points in RELLIS-3D. Because both datasets contain a similar distribution of points, any effects of class imbalance would be replicated across both datasets. Indeed, we see that on average, across both datasets, classes with more points have higher mIOU scores than classes with fewer points. This shows the importance of having a balanced dataset for the ability to detect all classes well. Because the disparity of class representation is replicated across datasets, we believe that class imbalance is not a significant contributor to the lower performance of the four algorithms on the RELLIS-3D dataset compared to the SemanticKITTI dataset.

4.4.3 Problems with RELLIS-3D labeled data

When analyzing the data from the output of the networks and the labeled data, we found several instances of inconsistencies in the labeled data from RELLIS-3D. In a significant number of scans throughout the dataset, there is a square of points in the center of the scan that have been labeled Void. There are other problems present throughout the RELLIS-3D dataset, including mislabels and switching labels. Different parts of large objects such as a person or large tree are regularly classified as more than one label in the same scan, i.e. a person's upper body is labeled as person and the lower body is labeled as grass. Sets of consecutive scans also have large groupings of points that swap back and forth between different labels, lacking consistency.

We would like to point out the inconsistency as a possible contributor to the poor performance of the networks on RELLIS-3D compared to their performance on SemanticKITTI. If the labels are inconsistent, it will weaken the ability for the network to learn which features belong to which class and lower the confidence of the network's predictions. In the worst case, the network could fail to distinguish between classes altogether. Additionally, since the inconsistencies belong to the published test set, the evaluation of points will be incorrect for any points that are labeled incorrectly. In a scenario of inconsistency with flipping labels, a label may swap from one class to another for several scans. If the network predictions do not swap with the inconsistent labels, those predictions will be counted as incorrect even though they are more consistent and presumably correct, lowering the mIOU score, due to poor labeling. It is impossible to know how many correctly predicted points are considered incorrect due to inaccurate labels. This inconsistency in labeling reduces the robustness and reliability of the dataset. These problems are discussed further and multiple examples are provided in the "Appendix".

4.5 Inference time results

Inference was performed on a laptop executing on Ubuntu 20.04 with an Intel i7-10750 H CPU, 16 GB of DDR4 RAM, and an external NVIDIA RTX A5000 GPU with 24 GB of VRAM. Table 7 shows inference time results for each algorithm which measures network prediction efficiency. The Ouster and Velodyne sensors used to collect the RELLIS-3D and SemanticKITTI datasets generally

Table 7 Inference time results

Network	Inference time (ms)
KPConv	29.97
SalsaNext	17.83
Cylinder3D	99.92
SphereFormer	94.01

output data at 10Hz. This means that an algorithm needs to execute in under 100ms to be considered real-time when running inference on every frame.

Cylinder3D had the highest mIOU score, but took the longest for inference; nearly 100 ms. SphereFormer was slightly faster, but not significantly. Both of these networks may be able to execute in real time, but would likely require powerful hardware and significant power consumption to do so.

SalsaNext was specifically designed to run in real time (Cortinhal et al. 2020). In our testing, it achieves a runtime more than five times faster than Cylinder3D, and almost twice as fast as the next fastest, KPConv. Sacrifices had to be made in raw mIOU performance, but the speed could compensate for this in some applications. When segmenting the most represented classes of Grass, Tree, Person, and Bush, SalsaNext had an mIOU score that was not significantly lower than the other algorithms. If the application does not require accurate segmentation of the less represented classes, such as an application using the segmentation to determine what areas are or are not traversable, SalsaNext would be a good choice, as it would save computation time and power for other tasks.

KPConv had a runtime of 29.97 ms. This is somewhat slower than SalsaNext, but still competitive compared to Cylinder3D and SphereFormer. As discussed in Sect. 5.4, we had to increase the size of the inference radius significantly to get KPConv to compute the entire point cloud. The run-time presented in this thesis was measured with a 50 m radius sphere, the largest that could fit in the 24 GB of VRAM in our laptop evaluation hardware. With the potential method of dividing the scan into 4 m radius spheres, as discussed in Sect. 5.4, the accuracy may improve, but the runtime would likely increase drastically. To cover the same area as a single sphere of radius 50 m, at least 1900 spheres of radius of 4 m would be needed. Running inference on a sphere of radius of 4 m took on average 14.42 ms. multiplying this by the minimum of 1900 spheres needed to cover the same area, we find a hypothetical time of more than 27,000 ms, or almost half of a minute, to run inference on a single scan.

5 Network analysis

In this section, we analyze the four algorithms evaluated in this work, presented in descending mIOU score on the RELLIS-3D dataset: Cylinder3D, SphereFormer, SalsaNext, and KPConv. We examine strengths and weaknesses of the algorithms and provide an analysis of the performance.

5.1 Cylinder3D

Cylinder3D had the highest mIOU score in the remaining eight classes and the highest overall score. Performance on the Pole, Vehicle, and Barrier classes is particularly impressive. The structure of each of these classes and the Person class, on which Cylinder3D also has the highest score, fits well with the design of the asymmetrical residual block in the Cylinder3D network. This residual block powers the convolutional kernel and allows the algorithm to focus more strongly on points in the immediate neighborhood of an object. Each of these classes has a distinct shape that is not surrounded by other points. On the rest of the classes except Log and Rubble, Cylinder3D performed well. Cylinder3D is the best overall algorithm we tested, although it has weaknesses in detecting certain types of objects.

5.2 SphereFormer

SphereFormer was the second best performing algorithm, with the highest score for several under-represented classes such as Log, Fence, Mud, and Rubble. Although SphereFormer performed better than the other networks on these classes, they still had poor performance and could benefit from a greater representation in the dataset. On more common classes like Grass, Tree, Person, and Bush, SphereFormer performed well with an mIOU score only slightly lower than the best.

Self-attention and exponential splitting enabled SphereFormer to achieve good overall performance. Self-attention allowed the algorithm to focus on similar point structures from across the whole scan (Matteazzi et al. 2024). Exponential splitting allowed for more distributed points to be considered by the convolution kernel at the same time. We performed an ablation study by removing the exponential splitting function from the network and retraining for 50 epochs using the same method as described in Sect. 3.3. Evaluating the modified model gave an overall decrease in mIOU score of 14.01%. Every class also had a decrease in performance, except Person with a 1.5% gain. All instances of Person in the dataset are located close to the sensor. Because of this, there is no benefit to having larger, more distributed kernels for classes, like Person, that are always

clustered around the sensor. This shows that exponential splitting improves performance on classes with distributed points, while negatively impacting classes with points biased to the center.

5.3 SalsaNext

SalsaNext did not have the highest mIOU score for any of the classes and had the worst score for Tree, Pole, and Bush. On these three classes, it was only 6%, 7%, and 1% worse than KPConv. It also performed very poorly on the Log, Fence, and Barrier classes. SalsaNext removed the strided convolution SalsaNet used for downsampling and instead replaced it with average pooling. The creators of SalsaNext hypothesized that learning at that level was not needed and wanted to reduce the number of trainable parameters to increase the network's speed. Other works have identified that implementing strided convolution over pooling for downsampling can lead to a more expressive model that boosts accuracy (Springenberg et al. 2015). The classes SalsaNext performed the worst on were ones with small features indicating that the average pooling downsampling may be impacting the performance.

On most of the other classes, SalsaNext had good performance, closer to the high mIOU scores of SphereFormer and Cylinder3D than the low scores of KPConv. Although not the best at any one class or overall, SalsaNext achieves its goal of fast runtime, as shown in Sect. 4.5. The trade-off between raw mIOU score and inference time makes SalsaNext a strong contender for the best choice in many applications.

5.4 KPConv

KPConv consistently performed poorly. This could be an artifact of the evaluation method, as we were unable to find a good evaluation script. The way that KPConv was trained and evaluated by default was for each iteration to randomly sample a point from within the entire dataset. A convolutional kernel is applied to a 4 m radius sphere surrounding that point. During training, the algorithm learned based on the points within the 4 m sphere. According to the author in a GitHub issue (https://github.com/HuguesTHOMAS/KPConv-PyTorch/issues/191), this was done to allow the algorithm to fit within a reasonable amount of GPU memory. Indeed, we saw this issue when testing, as larger radii used more GPU memory.

For evaluation, we tried two different approaches and suggest a possible third approach. The first approach was to simply run the evaluation script included in the KPConv GitHub repository with its default parameters. This approach runs 100 epochs of testing and compiles all predicted points throughout the evaluation process. It takes several hours to complete, and even with 10,000 4 m radius spheres sampled per epoch and 100 epochs, it still did not sample every point in the dataset, resulting in a large number of black Void points in scans.

The second approach, which we used to calculate the results presented in Table 4, was to evaluate the entire point cloud at once by changing the radius of the convolution kernel sphere to 50 m. This captured the entire point cloud in one step and avoided missing large sections due to random sampling. Every point in an individual scan was labeled in one inference iteration.

Using these two approaches, mIOU scores ranged from 17 to 43. This variance in mIOU values was directly attributable to the random sampling of the algorithm, the size of the convolutional sphere, and the overall number of points considered in the evaluation. We admit that changing the sphere size is a flawed approach, but we believe that it is the best option. When using a machine learning algorithm to perform semantic segmentation, it is most useful to receive a prediction of the whole image or scene, rather than a small select portion. Therefore, the most useful evaluation is one that infers over all data in a single scan. Another possible evaluation method might be to divide the entire point cloud into 4 m spheres such that every point is contained in a sphere. This would enable evaluation in a more similar manner to the training process, but it would be computationally impractical for real-time execution, as discussed in Sect. 4.5.

6 Conclusion

Each of the four algorithms evaluated, KPConv, SalsaNext, Cylinder3D, and SphereFormer had a lower mIOU score on the RELLIS-3D dataset than the SemanticKITTI dataset. We believe this is due to a variety of factors. The largest factor could be inconsistencies in the labels. In a small sampling of the dataset, 71% of scans had at least some points that were obviously labeled incorrectly. This directly affects evaluation, as correctly predicted points evaluated against incorrect labels will artificially decrease the mIOU score. Incorrect and inconsistent labels will also affect the training of the algorithms, but it is impossible to predict the exact impacts without relabeling the entire dataset.

The next factor in the lower score when compared to SemanticKITTI is the inherent difficulty of segmenting unstructured environments. It was impossible for us to determine the boundaries between some classes in the raw data. This difficulty extends to the labeling procedure and to network inference.

Finally, class imbalance was also a factor in the performance of each algorithm on RELLIS-3D, with the classes that were better represented performing better on average than the ones with fewer points. All algorithms had difficulty

when running inference on under-represented classes such as Rubble, Log, Fence, and Mud. This imbalance was replicated in the SemanticKITTI dataset, however. Any decrease in performance due to class imbalance should be replicated across both datasets. More points of under-represented classes are needed to improve the performance of these algorithms on both datasets.

This research shows some of the strengths and weaknesses of each of the four networks on an unstructured dataset. SphereFormer and Cylinder3D both worked very well for most classes, with Cylinder3D performing slightly better overall. SphereFormer was the best at SemanticKITTI, when it was provided with more data, while Cylinder3D did slightly better with the more limited dataset of RELLIS-3D. SalsaNext has a fast inference time and is acceptable at detecting most classes well represented in the dataset. KPConv had poor results due to difficulties in running inference, but may be useful in limited scenarios.

Qualitatively, we showed that most of the networks, especially Cylinder3D and SphereFormer, were able to generalize and perform well on common classes. Shockingly, the predictions from these algorithms are more consistent than the human-generated labels. Segmenting traversable and un-traversable classes remains a core challenge for automating exploration of unstructured environments with AGVs.

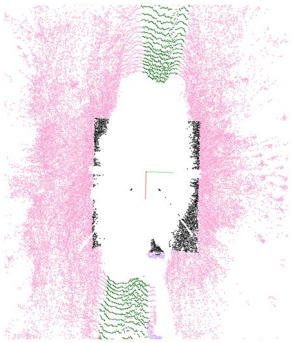

Fig. 7 Square of void (labeled in black)

Appendix: RELLIS-3D labeled data problems

To quantify the number of scans that have this square of points labeled Void shown in Fig. 7, we randomly sampled three sequences of 100 scans from the dataset. Of the 300 scans sampled, the square of Void points was present in 216 of them, or 72%. The center of each scan commonly contains points that belong to the Person class, since there was a driver and three other people following the AGV throughout the data collection. These four people are present in every scan, but are sometimes labeled Void, shown in Black, and sometimes labeled Person when they are sufficiently far from the sensor, and sometimes split between the two when on the threshold. Additionally, other points around the AGV are also labeled Void. These are commonly surrounded by points labeled Grass or Bush. We suspect that the Void label is inaccurate and an artifact of the labeling process.

There are several locations in the RELLIS-3D dataset where the points in the same location flip labels in successive scans. Examples of this have been provided in Figs. 8, 9, and 10. Figure 8 shows a sequence of three consecutive scans. The points in the top center of the scans are labeled Grass, shown in dark green, on the left in scan 2468. A large number of points swap to Bush in scan 2469, and then back to Grass in scan 2470.

Figure 9 shows two consecutive scans that illustrate a turning point. There is a sudden switch in the labeled class of a large area of the scan. The upper right quarter of the scan switches class from Bush to Grass in a single scan. In the scans prior to this, the area in question is consistently labeled Bush, although with some small inconsistency in

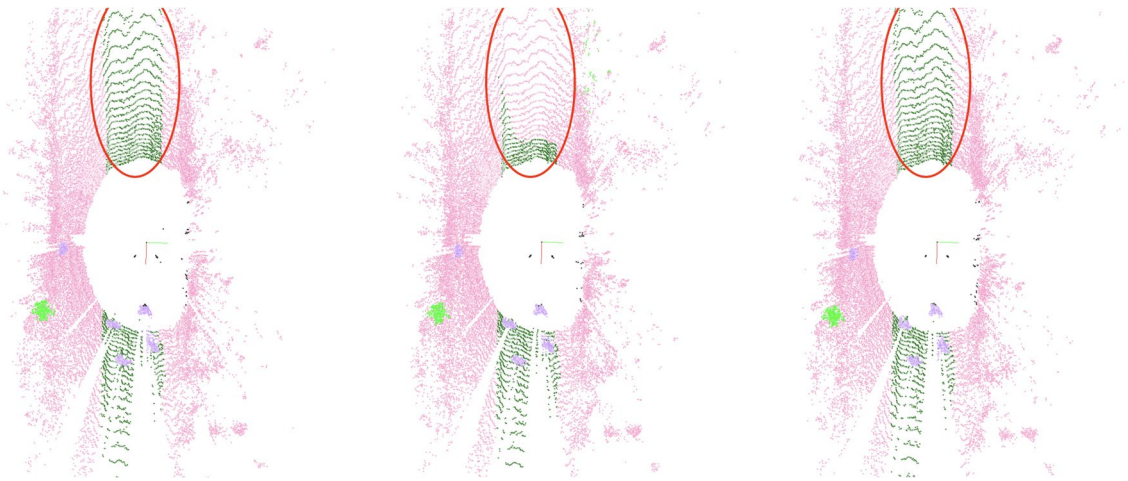

Fig. 8 Grass labels flipping class in Sequence 2

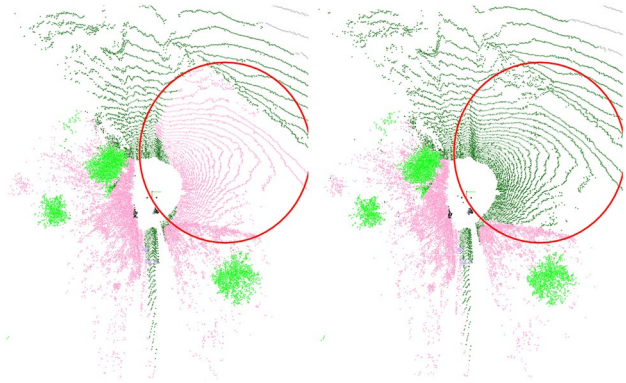

Fig. 9 Large section of points flips class in Sequence 2

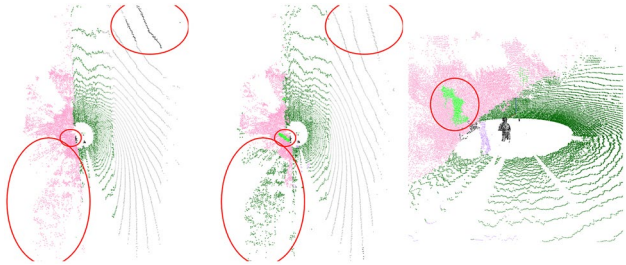

Fig. 10 Inconsistent labels in several classes in sequence 2

the boundary, like what was shown in Fig. 8. After the first scan in Fig. 9, the points are consistently labeled Grass, as shown in the second half of the screenshot.

These problems are shown again in Fig. 10, but with the addition of Void and Tree to the classes in question. In the first scan, the entire left half of the point cloud is labeled Bush, shown in pink and there are some points in the top right corner labeled Void, shown in black. In the second scan, a large section of points have been labeled Grass, shown in green and a smaller section labeled Tree, shown in light green, and the location where the Void points were is now labeled Concrete, shown in grey. The last figure shows an isometric view of the second scan, but zoomed in. We can see the person in the middle of the scan is labeled Void while the person behind is labeled Person, shown in purple. There are also points labeled Tree where there were not in the previous scan.

To attempt to quantify the amount of scans that were labeled incorrectly, we performed manual evaluation of three randomly chosen sequences of 100 scans from the dataset. For each of these sequences, we counted the number of obvious inconsistencies, such as groups of points flipping between classes, like in Fig. 8, points in the middle of a class that were labeled as another, or the presence of the square of Void points in the center of the scan, as shown in Fig. 7 . We report these results in a binary manner; either there are inconsistencies, or there are not. In reality, there is a spectrum of inconsistent points. Some scans had several dozen points that were obviously incorrect, either as a patch of points like the Void seen in the top right corner or the Tree in Fig. 10. Others had larger patches of incorrect points, like in Fig. 9.

Upon visual inspection, we found that 124 out of 300 scans had obvious inconsistencies, excluding the square of Void at the center of the scan. 216 out of the same 300 scans were also affected by the Void points. This is 41% and 72% of scans, respectively. Some classes will be impacted by the inconsistencies more than others. Mislabeled Vehicle points will have a greater impact than mislabeled Grass points due to the larger number of total points in the Grass class.

The numbers presented here are unlikely to be representative of the entire dataset, but are meant to provide a rough estimate of possible impact. Additionally, exact values for the numbers and percentages of incorrectly labeled points are impossible to accurately determine without relabeling the scans and evaluating the difference. Due to the difficulty in determining classes of points without a-priori knowledge of the environment, as discussed in Sect. 4.4.1, we will not attempt this, as our estimates of classes may be as bad or worse than the existing labels. However, the impact of incorrect labels cannot be ignored. For context, following a similar evaluation methodology, we reviewed hundreds of scans in the SemanticKITTI dataset without a single instance of dynamically changing labels.

Acknowledgements We would like to acknowledge the Electrical and Computer Engineering department at the University of Alabama for their support.

Authors' contributions Conceptualization: M.M., L.E., Y.Y., and K.R.; Data preparation: M.M. and L.E.; Methodology: M.M.; Formal analysis and investigation: M.M., L.E., and Y.Y.; Writing - original draft preparation: M.M. and L.E.; Writing - review and editing: M.M., L.E., Y.Y., and K.R.; Supervision: K.R.

Funding No funding was received to assist with the preparation of this manuscript.

Data availibility No datasets were generated or analysed during the current study.

Code availability All code is publicly available at https://github.com/UA-Lidar-Segmentation-Research.

Declarations

Conflict of interest All authors certify that they have no affiliations with or involvement in any organization or entity with any financial interest or non-financial interest in the subject matter or materials discussed in this manuscript.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

Aksoy, E.E., Baci, S., Cavdar, S.: SalsaNet: Fast Road and Vehicle Segmentation in LiDAR Point Clouds for Autonomous Driving. arXiv (2019). https://doi.org/10.48550/arXiv.1909.08291

Behley, J., Garbade, M., Milioto, A., Quenzel, J., Behnke, S., Stachniss, C., Gall, J.: SemanticKITTI: a dataset for semantic scene understanding of LiDAR sequences. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 9296–9306. IEEE, Seoul (2019). https://doi.org/10.1109/ICCV.2019.00939

Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuScenes: A Multimodal Dataset for Autonomous Driving. arXiv (2020). https://doi.org/10.48550/arXiv.1903.11027

Cortinhal, T., Tzelepis, G., Erdal Aksoy, E.: SalsaNext: fast, uncertainty-aware semantic segmentation of LiDAR point clouds. In: Bebis, G., Yin, Z., Kim, E., Bender, J., Subr, K., Kwon, B.C., Zhao, J., Kalkofen, D., Baciu, G. (eds.) Advances in Visual Computing. Lecture Notes in Computer Science, pp. 207–222. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-64559-5_16

Dimensionality, Potential-based sampling, input spheres and batch_ neighbors in Classification. Issue #191. HuguesTHOMAS/KPConv-PyTorch. https://github.com/HuguesTHOMAS/KPConv-PyTorch/issues/191

Eastepp, M., Faris, L., Ricks, K.: UA_l-DoTT: University of Alabama's large dataset of trains and trucks. Data in Brief **42**, 108073 (2022). https://doi.org/10.1016/j.dib.2022.108073

Felter, W., Ferreira, A., Rajamony, R., Rubio, J.: An updated performance comparison of virtual machines and Linux containers. In: 2015 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), pp. 171–172 (2015). https://doi.org/10.1109/ISPASS.2015.7095802

Fong, W.K., Mohan, R., Hurtado, J.V., Zhou, L., Caesar, H., Beijbom, O., Valada, A.: Panoptic nuScenes: A Large-Scale Benchmark for LiDAR Panoptic Segmentation and Tracking. arXiv (2021). https://doi.org/10.48550/arXiv.2109.03805

Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? The KITTI vision benchmark suite. In: 2012 IEEE Conference on Computer Vision and Pattern Recognition, pp. 3354–3361 (2012). https://doi.org/10.1109/CVPR.2012.6248074

Geyer, J., Kassahun, Y., Mahmudi, M., Ricou, X., Durgesh, R., Chung, A.S., Hauswald, L., Pham, V.H., Mühlegg, M., Dorn, S., Fernandez, T., Jänicke, M., Mirashi, S., Savani, C., Sturm, M., Vorobiov, O., Oelker, M., Garreis, S., Schuberth, P.: A2D2: Audi Autonomous Driving Dataset. arXiv (2020). https://doi.org/10.48550/arXiv.2004.06320

Graham, B., Maaten, L.: Submanifold Sparse Convolutional Networks. arXiv (2017). https://doi.org/10.48550/arXiv.1706.01307

Graham, B., Engelcke, M., Maaten, L.: 3D Semantic Segmentation with Submanifold Sparse Convolutional Networks. arXiv (2017). https://doi.org/10.48550/arXiv.1711.10275

Guo, M.-H., Cai, J.-X., Liu, Z.-N., Mu, T.-J., Martin, R.R., Hu, S.-M.: PCT: point cloud transformer. Comput. Vis. Med. **7**(2), 187–199 (2021). https://doi.org/10.1007/s41095-021-0229-5

Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N., Markham, A.: RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds, pp. 11105–11114. IEEE Computer Society, Seattle (2020). https://doi.org/10.1109/CVPR42600.2020.01112

Huang, X., Wang, P., Cheng, X., Zhou, D., Geng, Q., Yang, R.: The ApolloScape open dataset for autonomous driving and its application. IEEE Trans. Pattern Anal. Mach. Intell. **42**(10), 2702–2719 (2020). https://doi.org/10.1109/TPAMI.2019.2926463

Jiang, P., Osteen, P., Wigness, M., Saripalli, S.: RELLIS-3D Dataset: Data, Benchmarks and Analysis. arXiv (2022). https://doi.org/10.48550/arXiv.2011.12954

Kong, L., Liu, Y., Chen, R., Ma, Y., Zhu, X., Li, Y., Hou, Y., Qiao, Y., Liu, Z.: Rethinking Range View Representation for LiDAR Segmentation. arXiv (2023). https://doi.org/10.48550/arXiv.2303.05367

Kopacek, P.: Robots for Humanitarian demining. In: Voicu, M. (ed.) Advances in Automatic Control. The Springer International Series in Engineering and Computer Science, pp. 159–172. Springer (2004). https://doi.org/10.1007/978-1-4419-9184-3_11

Kushwaha, H., Sinha, J.P., Khura, T., Kushwaha, D., Ekka, U., Purushottam, M., Singh, N.: Status and scope of robotics In Agriculture. In: International Conference on Emerging Technologies in Agricultural and Food Engineering, Kharagpur (2016)

Lai, X., Chen, Y., Lu, F., Liu, J., Jia, J.: Spherical Transformer for LiDAR-based 3D Recognition. arXiv (2023). https://doi.org/10.48550/arXiv.2303.12766

Li, J., Zhang, C., Cao, Q., Qi, C., Huang, J., Xie, C.: An experimental study on deep learning based on different hardware configurations. In: 2017 International Conference on Networking, Architecture, and Storage (NAS), pp. 1–6. IEEE, Shenzhen (2017). https://doi.org/10.1109/NAS.2017.8026843

Li, Y., Ma, L., Zhong, Z., Liu, F., Cao, D., Li, J., Chapman, M.A.: Deep Learning for LiDAR Point Clouds in Autonomous Driving: A Review. arXiv (2020). https://doi.org/10.48550/arXiv.2005.09830

Liu, Z., Zhao, X., Huang, T., Hu, R., Zhou, Y., Bai, X.: TANet: Robust 3D Object Detection from Point Clouds with Triple Attention. arXiv (2019). https://doi.org/10.48550/arXiv.1912.05163

Matteazzi, A., Colling, P., Arnold, M., Tutsch, D.: A preprocessing and Postprocessing Voxel-Based Method for Lidar Semantic Segmentation Improvement in Long Distance, Germany (2024). https://doi.org/10.48550/arXiv.2405.10046

Murphy, R.: Disaster Robotics. Intelligent Robotics and Autonomous Agents. Mit Pr, USA (2014). https://doi.org/10.7551/mitpress/9407.001.0001

Nagatani, K., Kiribayashi, S., Okada, Y., Otake, K., Yoshida, K., Tadokoro, S., Nishimura, T., Yoshida, T., Koyanagi, E., Fukushima, M., Kawatsuma, S.: Emergency response to the nuclear accident at the Fukushima Daiichi Nuclear Power Plants using mobile rescue robots. J. Field Robot. **30**(1), 44–63 (2013). https://doi.org/10.1002/rob.21439

Papers with Code—The latest in Machine Learning. https://paperswithcode.com/

Ronneberger, O., Fischer, P., Brox, T.: U-Net: Convolutional Networks for Biomedical Image Segmentation. arXiv (2015). https://doi.org/10.48550/arXiv.1505.04597

Roynard, X., Deschaud, J.-E., Goulette, F.: Paris-Lille-3D: a large and high-quality ground truth urban point cloud dataset for automatic segmentation and classification. arXiv (2018). https://doi.org/10.48550/arXiv.1712.00032

Springenberg, J.T., Dosovitskiy, A., Brox, T., Riedmiller, M.: Striving for Simplicity: The All Convolutional Net, Germany (2015). https://doi.org/10.48550/arXiv.1412.6806

Sun, P., Kretzschmar, H., Dotiwalla, X., Chouard, A., Patnaik, V., Tsui, P., Guo, J., Zhou, Y., Chai, Y., Caine, B., Vasudevan, V., Han, W., Ngiam, J., Zhao, H., Timofeev, A., Ettinger, S., Krivokon, M., Gao, A., Joshi, A., Zhao, S., Cheng, S., Zhang, Y., Shlens, J., Chen, Z., Anguelov, D.: Scalability in Perception for Autonomous Driving: Waymo Open Dataset. arXiv (2020). https://doi.org/10.48550/arXiv.1912.04838

Tang, H., Liu, Z., Zhao, S., Lin, Y., Lin, J., Wang, H., Han, S.: Searching Efficient 3D Architectures with Sparse Point-Voxel Convolution. arXiv (2020). https://doi.org/10.48550/arXiv.2007.16100

Testolina, P., Barbato, F., Michieli, U., Giordani, M., Zanuttigh, P., Zorzi, M.: SELMA: SEmantic large-scale multimodal acquisitions in variable weather, daytime and viewpoints. IEEE Trans. Intell. Transp. Syst. **24**(7), 7012–7024 (2023). https://doi.org/10.1109/TITS.2023.3257086

Thomas, H., Qi, C.R., Deschaud, J.-E., Marcotegui, B., Goulette, F., Guibas, L.: KPConv: flexible and deformable convolution for point clouds. In: 2019 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 6410–6419 (2019). https://doi.org/10.1109/ICCV.2019.00651

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention Is All You Need. arXiv (2023). https://doi.org/10.48550/arXiv.1706.03762

Wang, P.: Research on comparison of LiDAR and camera in autonomous driving. J. Phys. Conf. Ser. **2093**(1), 012032 (2021). https://doi.org/10.1088/1742-6596/2093/1/012032

Wigness, M., Eum, S., Rogers, J.G., Han, D., Kwon, H.: A RUGD dataset for autonomous navigation and visual perception in unstructured outdoor environments. In: 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 5000–5007. IEEE, Macau (2019). https://doi.org/10.1109/IROS40897.2019.8968283

Yan, X., Gao, J., Zheng, C., Zheng, C., Zhang, R., Cui, S., Li, Z.: 2DPASS: 2D Priors Assisted Semantic Segmentation on LiDAR Point Clouds. arXiv (2022). https://doi.org/10.48550/arXiv.2207.04397

Zhu, X., Zhou, H., Wang, T., Hong, F., Li, W., Ma, Y., Li, H., Yang, R., Lin, D.: Cylindrical and Asymmetrical 3D Convolution Networks for LiDAR-based Perception. arXiv (2021). https://doi.org/10.48550/arXiv.2109.05441

Mason McVicker received his B.S. in Computer Engineering and M.S. in Electrical Engineering from the University of Alabama in 2021 and 2023 respectively. His research interests include mobile robotics, lidar processing, deep learning, and autonomous navigation.

Lauren Ervin received her B.S. in Electrical Engineering with a Computer Option from The University of Alabama in 2020. She is now pursuing a Ph.D. from the same department. She is a 2022–2025 NASA ASGC fellow, 2023 NSF BPart fellow, and 2024 NSF iREDEFINE fellow. Her research interests include mobile robotics designed for space applications, unstructured environment traversal, computer vision, deep learning, and dynamic modeling.

Yongzhi Yang received his B.S. in Electrical Engineering from Linyi University in 2015. He is now pursuing a Ph.D. in Electrical and Computer Engineering from The University of Alabama. His research interests include deep learning, 3D image processing, unstructured environment autonomous navigation, real-time negative obstacle avoidance.

Dr. Kenneth G. Ricks received the B.S. degree in Electrical Engineering from the University of Alabama, Tuscaloosa in 1989, and the M.S. and Ph.D. degrees from the University of Alabama in Huntsville in 1997 and 2002, respectively. From 1989–2002 he worked at the National Aeronautics and Space Administration (NASA) Marshall Space Flight Center in Huntsville, Alabama. Since 2002, Dr. Ricks has worked in the Department of Electrical and Computer Engineering at the University of Alabama where he is currently a Professor and the Interim Department Head. Dr. Ricks' research interests include embedded systems, real-time computation, computer architecture, robotics, autonomous navigation, and artificial intelligence. He is a Senior Member of IEEE.