AI-based landing zone detection for vertical takeoff and land LiDAR localization and mapping pipelines

Balasooriya, Narmada M.; De Silva, Oscar; Jayasiri, Awantha; Mann, George K.I.

Publisher's version / Version de l'éditeur:
https://doi.org/10.1139/dsa-2022-0038
Drone Systems and Applications, 12, pp. 1-19, 2024-04-02

Drone Systems and Applications

OPEN ACCESS | Article

AI-based landing zone detection for vertical takeoff and land LiDAR localization and mapping pipelines

Narmada M. Balasooriya[a], **Oscar De Silva**[a], **Awantha Jayasiri**[b], **and George K.I. Mann**[a]

[a]Memorial University of Newfoundland, Canada; [b]National Research Council of Canada Flight Research Laboratory, Canada

Corresponding author: **Narmada M. Balasooriya** (email: balasooriyab@mun.ca)

Abstract

This work describes a deep learning-based autonomous landing zone identification module for a vertical takeoff and landing vehicle. The proposed module is developed using LiDAR point cloud data and can be integrated into a visual LiDAR odometry and mapping pipeline implemented in the vehicle. "ConvPoint," the top-performing neural network architecture in an online point cloud segmentation benchmark leaderboard at the time of writing, was chosen as the reference architecture. Semantic labeling of the datasets was done using the terrain geometry characteristics and manual adjustment of labels through visual observation. Point clouds captured by the Memorial University and online point cloud datasets were used to transfer-learn the neural network model and to evaluate the accuracy-runtime trade-off for the proposed pipeline. The selected neural network model generated accuracy values of 89.7% and 92.1% on two selected datasets, while it computed 3940.15 points per second and 3633.85 points per second to predict landing zone labels, respectively. Hyperparameter tuning was carried out to obtain a higher throughput with an update rate of 1 Hz for the landing zone map of the point cloud inputs from the visual LiDAR odometry and mapping pipeline. The proposed system is validated by evaluating its performance on three variations of point clouds. The results validate the accuracy-runtime trade-off of the proposed system and show that further optimization can improve performance.

Key words: vertical take-off and landing vehicles, landing zone detection, LiDAR, deep learning, neural networks, VLOAM

1. Introduction

Unmanned aerial vehicles (UAVs) have become popular in several application domains, especially multi-rotor vertical take-off and landing (VTOL) vehicles with the capability to operate in constrained and unstructured environments. VTOL autonomous capabilities are incredibly useful in hazardous and hard-to-reach environments (Scherer et al. 2012), where the technology has the capability of enhancing the partial and full authority cases with informative pilot displays for safe and accurate operation. During critical operations, UAVs should be able to land safely without compromising the safety of the personnel involved and the surrounding infrastructure. Although the existing commercial drones have the functionality to take off and land without the operator's command, choosing a terrain to land is primarily a decision made by the operator (Scherer et al. 2012). Equipping these vehicles with the capability to identify safe landing zones (LZs) will enable VTOLs to operate autonomously in unknown environments, ensuring the overall safety of the autonomous navigation system, as well as reducing the pilot's workload.

Autonomous identification of safe LZs by a VTOL is a challenging task. Over the years, researchers have tried to introduce different approaches to tackle this challenge using various sensor data. It is necessary to understand that a safe LZ should be a low gradient, obstacle-free, open support surface with sufficient area to land the VTOL vehicles (Scherer et al. 2012). Classical approaches use mathematical modeling and traditional algorithms without machine learning to evaluate the LZs based on the slope, roughness, and ground features extracted from LiDAR and image data. In Desaraju et al. (2015), Warren et al. (2015), Kaljahi et al. (2019), and Lusk et al. (2019), authors use conventional image processing methods to detect potential LZs using the image data captured by onboard cameras. In Scherer et al. (2012), Garg et al. (2015), Lorenzo et al. (2016), and Yan et al. (2020), the authors use geometric computation on LiDAR data to detect and label the safe LZs. Even though these methods generated reasonable results on their chosen test data, due to the rule-based nature of the algorithms, they lack the ability to learn and adapt to new data that become available for LZ detection purposes.

Neural network-based LZ detection is a recent development in the domain, which offers several advantages over classical methods. Neural networks (NNs) can learn complex patterns related to LZ identification using labeled training data provided that there is sufficient quantity and diversity in training data representative of operational conditions. Additionally, NN-based methods have the ability to incorporate several different sources of data, such as LiDAR and images, to extract other critical information needed for LZ evaluation. Furthermore, NN architectures allow fast execution through

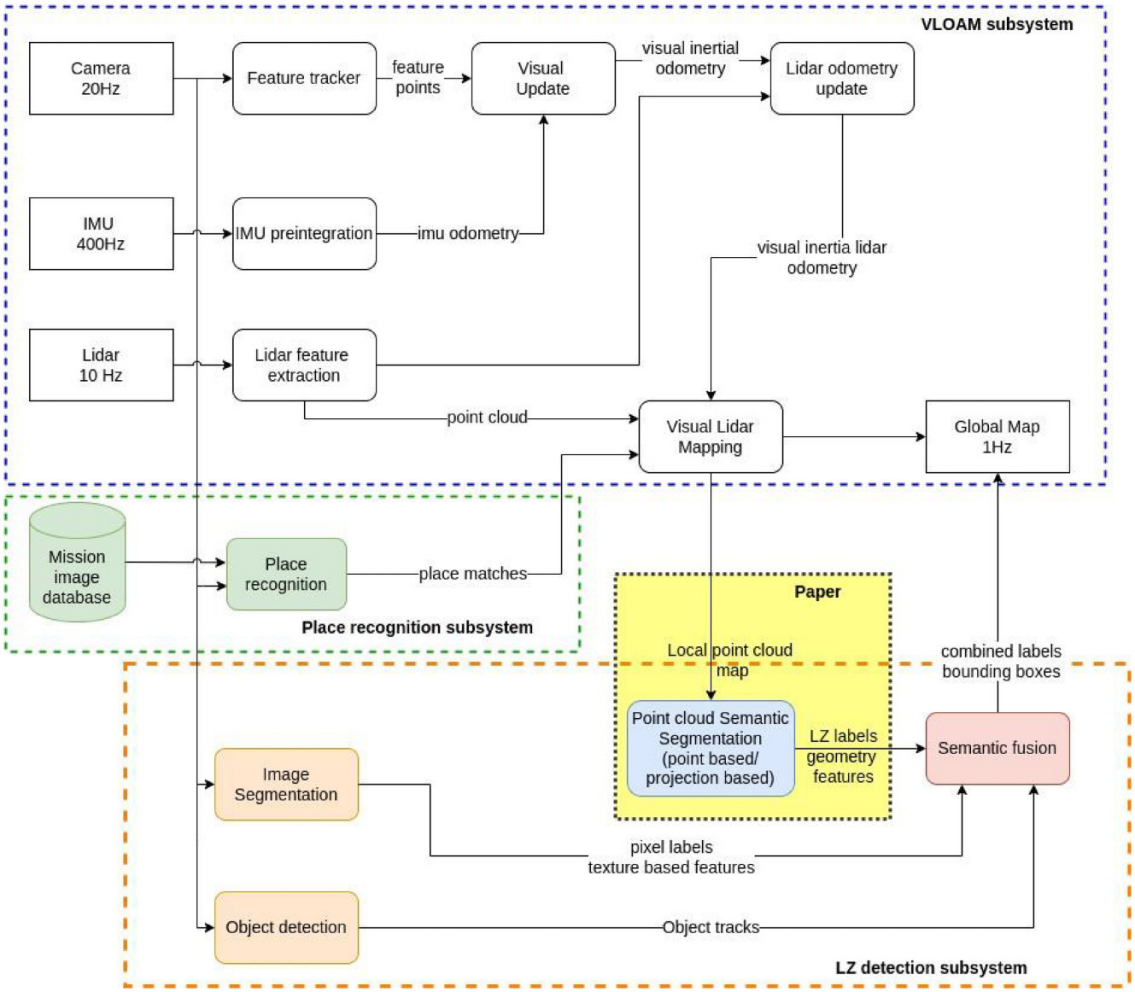
the use of graphics processing unit (GPU) acceleration using new machine learning libraries such as TensorFlow and PyTorch. Authors of Lee et al. (2020), Polvara et al. (2019), and Lee and Kwon (2020) developed computer vision-based convolutional neural network (CNN) algorithms for image segmentation, object detection, and tracking to detect possible safe LZs. Although CNN-based object detection and classification is a well-established area, vision data do not have the reliable depth information required to detect LZs accurately. Also, the inability to register data in poor lighting conditions, irregular sensor sensitivity, and limited field-of-view (FOV) make it difficult for the metric reconstruction of the environment using vision algorithms alone. Work in Maturana and Scherer (2015*a*) uses globally registered LiDAR point clouds with a predefined region of interest for landing sites to determine the probability of the safety prediction for each LZs. In their approach, a volumetric density map of the region of interest is fed into the CNN model to predict the safety of sub-volumes of the map. Authors of Maturana and Scherer (2015*b*) suggested a 3 D CNN architecture called "Voxnet" based on volumetric occupancy grid representation of point clouds. The 3 D convolution layers in their work are applied to voxel grids. They have implemented the Voxnet for various configurations, and the slowest configuration has taken 6 ms for classification by using a Tesla K40 GPU. However, voxelization of point clouds increases computational overhead, and the fine details of the original data can be lost in the process (Ye et al. 2018). Both Maturana and Scherer (2015a, 2015*b*) rely only on LiDAR data, which can also lead to inaccurately identifying surfaces like frozen lakes, water bodies, and unstable ground as LZs. Furthermore, for real-time LZ determination applications, the methods require compatibility and optimizations to operate with multi-sensor navigation and mapping pipelines.

Work in Long et al. (2015) shows how the deep NLs used for image classification can be adopted for pixel-wise semantic segmentation by utilizing convolutional layers. Semantic segmentation is the process of associating each pixel in an image or each point in a point cloud with a class label of choice. Theoretically, convolutional NLs should be able to process point clouds by extending the methods to a third dimension. The exponential increase in the memory requirement due to increased dimensionality makes this method computationally expensive for an onboard computer system for a VTOL. Different research was carried out to work on a solution to this memory issue and the problem of handcraft features in classical machine learning algorithms. This paved the way for the introduction of two main architectures of point cloud semantic segmentation, i.e., point-based and projection-based point cloud semantic segmentation networks. The point-based methods directly process 3 D points for semantic segmentation by utilizing 3 D convolutions, whereas the projection-based methods use a 2 D projection of the point cloud as the input and process it similarly to 2 D image semantic segmentation. The projection-based methods then map the class labels generated for the projection to the original point cloud to output the class labels for each point. Though point-based 3 D semantic segmentation methods deliver high accuracy and mean Intersection over Union (mIoU), they have higher computational costs compared to the projection-based methods.

PointNet (Qi et al. 2017*a*) and PointNet++ (Qi et al. 2017*b*) present the initial work on the point-based approach. Though these two methods are successful in delivering higher overall accuracy, they are too slow in processing large-scale point clouds. Additionally, the performance of the above two models in detecting vegetation, buildings, and vehicles does not meet the safety requirement of our problem in detecting safe LZs. RandLA-Net (Hu et al. 2021) uses a downsampling method to remove features at random and then a feature aggregation module to increase the receptive field for each 3 D point where geometric details are preserved. KPConv (Thomas et al. 2019) introduces a novel kernel point convolution method to process point clouds directly without intermediate computations. Though these RandLA-Net and KPConv methods deliver a promising performance, their architectures are not developed for a continuous input of point cloud data. Authors of Boulch (2020) propose an approach termed "ConvPoint" to process the unstructured point cloud data using a continuous convolution formulation. By this continuous convolution formulation, they designed NNs similar to 2 D CNNs to process the point cloud data as 3 D data. Their results prove the flexibility of this approach for different autonomous navigation architectures. Moreover, at the time of writing this article, the ConvPoint model had the highest accuracy and fastest execution for Semantic3D point cloud segmentation benchmark (Hackel et al. 2017*a*), which involves eight classes and supports both LiDAR and color data of the point cloud. According to the ConvPoint paper (Boulch 2020), the suggested model achieved an overall accuracy of 93.8% and an mIoU of 75.0% on the Semantic3D dataset. To address the lower speed of point-based methods, Wu et al. (2018) introduces a 2 D projection of a point cloud using spherical transformation and then applies 2 D semantic segmentation on the projections and the generated labels are reprojected onto the 3 D point cloud. Benefiting from these findings, the RangeNet++ (Milioto et al. 2019) model uses a DarkNet backbone to process a range-image generated by the projection of a point cloud. These methods produce a lower accuracy compared to point-based methods but are significantly faster in processing.

Although improvement in speed is possible with the projection-based network, the accuracy of these methods is still questionable, especially for safety-critical decisions like detecting LZs. Projection-based methods are designed to run on raw data, and point-based methods are developed to run on aggregated point cloud data, i.e., generated maps. However, for low-resolution LiDAR like the VLP-16 sensor, the projection-based methods also require a method of point cloud aggregation prior to range image generation for reasonable results. In safety-critical applications, high accuracy and precision in identifying safe LZs are more pertinent than the speed of the detection. This requirement overrules projection-based networks and demands a runtime optimization for point-based networks. We propose a novel LZ detection pipeline using point-based LiDAR semantic segmentation implemented with a visual LiDAR odometry and mapping (VLOAM) module to investigate accuracy-runtime

Fig. 1. Overview of the total architecture and the proposed LZ identification module in this paper (highlighted).



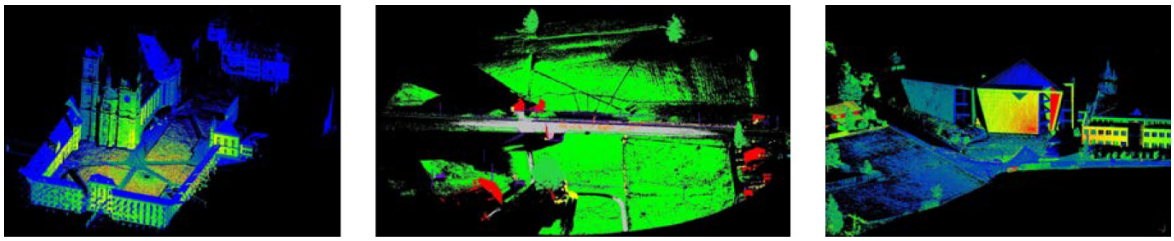
trade-offs for real-time applications. For the results of this paper, we only use image data within VLOAM as means of improving the navigation and mapping solution. Although image-based landing zone refinements are compatible with the pipeline, as discussed in Section 2.1, these image-based modules are not investigated in the scope of this paper. However, this paper will only focus on integrating a deep NN-based LZ detection module to a LiDAR mapping pipeline and investigating how point cloud aggregation or sub-mapping can be utilized to deliver an accuracy-runtime trade-off for online LZ detection. Application of deep NN for LZ detection has advantages of incremental and adaptive learning from new data, faster runtime, use of graphical processing unit for computations that can free the central processing unit (CPU) for other tasks, ease in fusing different NN networks, and code optimization capability compared to classical methods. To achieve this objective, we evaluate the top-performing point-based semantic segmentation model in the Semantic3D leaderboard for its performance on several variations of point cloud data. This paper presents the point cloud processing module, which is designed to be compatible with the overall architecture, and evaluates the capability of the system to provide an LZ proposal in a mapping pipeline. The results show how the proposed method can achieve the target accuracy-runtime trade-off when three variations of aerial LiDAR datasets are presented. The article also illustrates sample visualizations of how the NN model achieves the objective of LZ detection in a VLOAM navigation pipeline of a VTOL aircraft.

2. Method

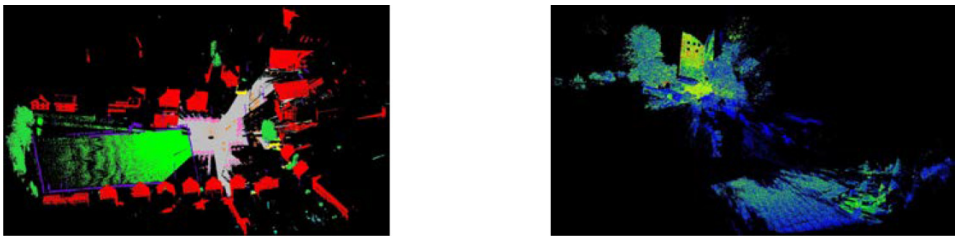
The use of 3 D convolution demands higher memory consumption, especially when larger point clouds or aggregated point cloud maps are processed. This inadvertently leads to higher processing times with high accuracy; hence, the selection should consider the accuracy speed trade-off for the real-time operational need. A point cloud processing pipeline is developed to meet the operational demands of VTOL applications, which is the ultimate objective of our proposed architecture.

This section will discuss the point cloud processing module, datasets, model selection, transfer-learning process, and performance metrics.

Fig. 2. Semantic3D datasets.

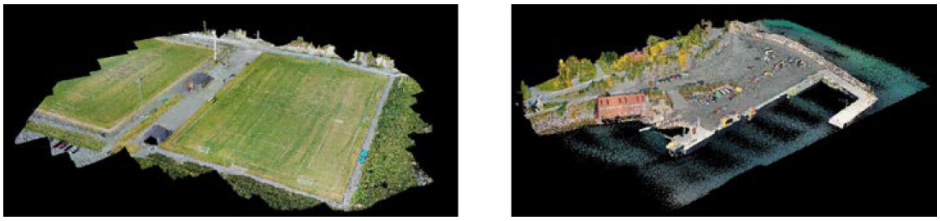


(a) Cathedral1 (b) sg27-station1 (c) sg28-station5



(d) sg27-station9 (e) Castle1

Fig. 3. Intelligent Systems Lab of the Memorial University of Newfoundland collected datasets.



(a) Paradise (b) Holyrood

Table 1. Number of points in the originally captured dataset versus downsampled dataset and the bounding box dimensions of the downsampled set.

Dataset	No.of points in original set	No.of points in downsampled set[a]	Dimensions in metres (x, y, and z)
Paradise	105 005 239	228 673	200.5, 189.0, and 29.635
Holyrood	37 948 660	450 850	176.875, 296.5, and 128.255

[a]The original large point cloud was downsampled using a voxelization method. The objective of this approach is to allow the point clouds to be processed by the selected models within the given memory constraints.

Fig. 4. Segmentation of Holyrood test set into sub-point clouds. Blue regions are non-landing zones (LZs) and the red regions are LZs.

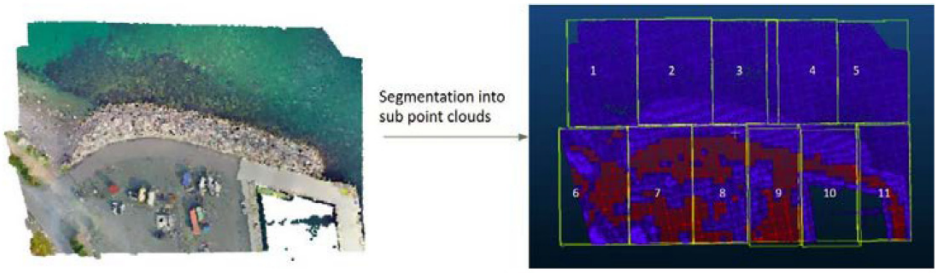
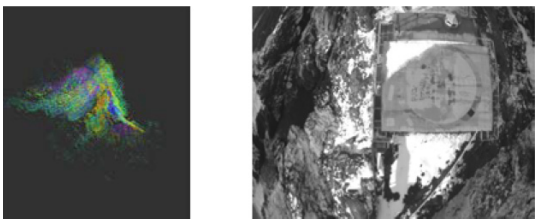


Table 2. Number of points in sub-point clouds.

Sub cloud	No. of points	Batch sizes[a]	Sampling sizes[b]
Sub-point cloud 1	9065		
Sub-point cloud 2	15793		
Sub-point cloud 3	11632		
Sub-point cloud 4	9937		
Sub-point cloud 5	8978		
Sub-point cloud 6	15 965	8, 16, 32, and 64	3000, 4000, 5000,..., and 9000
Sub-point cloud 7	20 436		
Sub-point cloud 8	15 365		
Sub-point cloud 9	15 319		
Sub-point cloud 10	15 187		
Sub-point cloud 11	13 600		

[a]The batch size parameter is changed to evaluate the runtime-accuracy tradeoff for each sub-point cloud.
[b]The sampling size is the number of points taken from the sub-point cloud in each batch processing.

Fig. 5. Visual LiDAR odometry and mapping (VLOAM) pipeline data.



(a) point cloud of the VLOAM pipeline data (b) An aerial image of the location

2.1. Point cloud processing module

VLOAM (Zhang and Singh 2015) is an odometry and mapping method that combines visual and LiDAR sensor data to create an accurate and reliable localization and mapping solution with a 6 degree-of-freedom (DoF) ego-motion estimation and a spatial representation of the environment. When visual sensors are used independently, they face challenges such as scale uncertainty with monocular cameras, sufficient lighting requirement, frame-to-frame lighting consistency, the need for static scenes, etc. The common challenges of LiDAR odometry are low data rate, the requirement of LiDAR pose information, and motion distortion due to scan rates. VLOAM architecture bridges over the weaknesses of both visual and LiDAR sensors allowing accurate estimation, mapping, and localization. This online odometry and mapping method simultaneously creates a point cloud map of the environment, while estimating the motion of the sensor system in that environment. In the VLOAM architecture, the visual odometry component accounts for motion estimation between two image frames, while the LiDAR odometry is responsible for transformation estimation and map construction from the registered scans.

In our proposed architecture, the point cloud semantic segmentation module is embedded into a VLOAM subsystem as shown in Fig. 1. In the VLOAM subsystem, there are three main sensor systems, i.e.,camera, inertial measurement unit (IMU), and LiDAR. Feature points tracked from the camera input and the IMU odometry computed by the IMU pre-integration are fed into the visual update unit, which computes the visual–inertial odometry solution. This visual–inertial odometry, combined with the extracted LiDAR features, is then input into the LiDAR odometry update unit to generate visual inertial LiDAR odometry. This output is then utilized to create a visual LiDAR mapping combined with the point cloud generated by the LiDAR unit. The place matches from the place recognition subsystem will also be used as an additional input into the visual Lidar mapping to make it more accurate and robust.

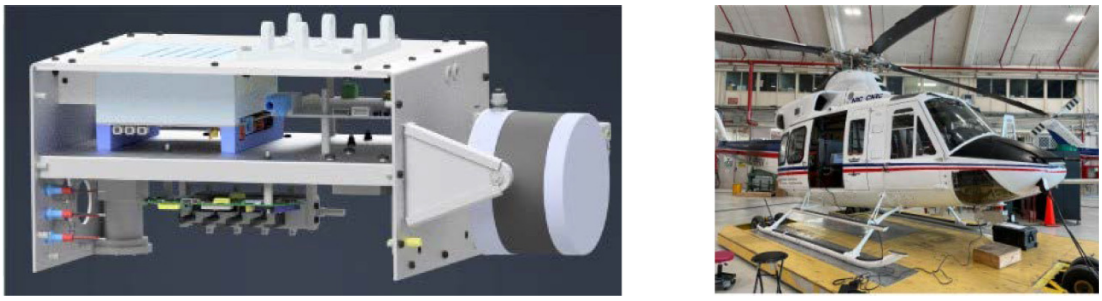
The LZ detection subsystem has a few components that will make use of the sensor units and the local point cloud map generated by the visual LiDAR mapping. The image segmentation and object detection modules are additional components that will assist in the LZ detection task by using texture-based features of the images. Both point cloud LZ labels and image pixel labels will be fused with object tracking to generate combined labels with bounding boxes for identified zones and objects.

As shown in Fig. 1, the VLOAM subsystem will generate a local point cloud map with a desired number of points that will be then published into the point cloud semantic segmentation module to generate the LZ labels. This paper concentrates on the LZ label generation by the point cloud semantic segmentation module and will detail the experiments on the module and the corresponding contribution to the whole system. The main objective of these experiments is to learn the best hyperparameter combination that can generate LZ labels from an NN model at a rate that can compete with the local point cloud map generation speed. A 1 Hz specified target is used for this study, which allows to incrementally color the generated point cloud by the VILOAM subsystem for LZ detection purposes.

2.2. Datasets

In this section, different compositions of datasets used to train and evaluate point cloud semantic segmentation will be discussed.

Fig. 6. Bell412 helicopter set-up with the payload.



(b) Bell-412 helicopter

Fig. 7. Flight path of Bell412 (Base Map Source: Google Earth).

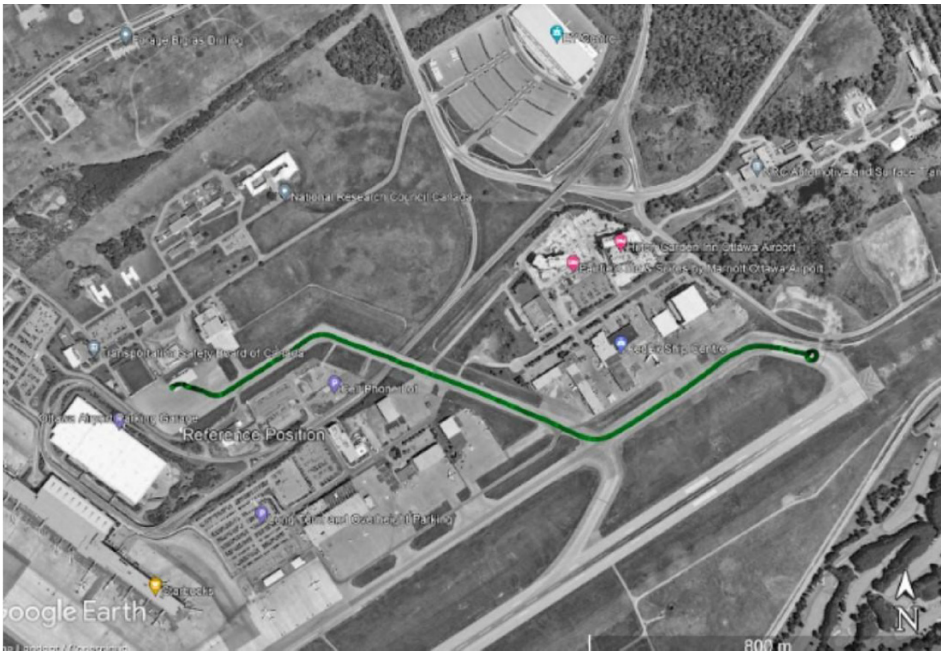
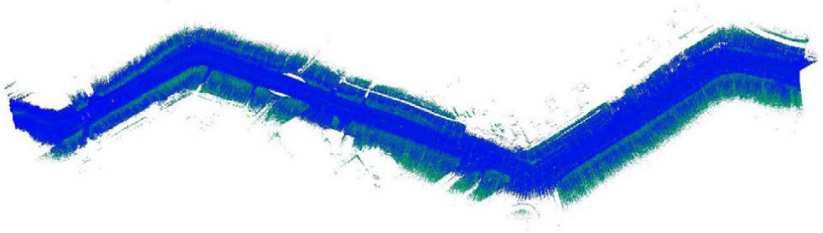


Fig. 8. Dataset 1 of visual LiDAR odometry and mapping pipeline data captured by Bell412 model.



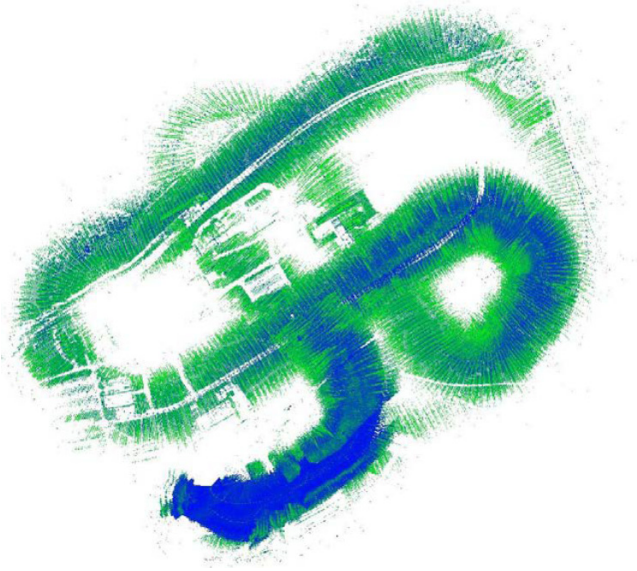
2.2.1. Semantic3D dataset

As the online dataset, the Semantic3D benchmark (Hackel et al. 2017*a*) was selected. This benchmark has 30 point clouds captured using a survey LiDAR on the ground, which are already categorized into eight classes, (1) man-made terrain, (2) natural terrain, (3) high vegetation, (4) low vegetation, (5) buildings, (6) hard scape, (7) scanning artifacts, (8) cars, and an additional label, (0) unlabeled points for data points without ground truth. Each data point consists of x, y, and z coordinates, intensity, and image RGB (red, green, and blue intensity) values. Five datasets, namely Cathedral1, sg27-station1, sg27-station9, sg28-station5, and Castle1, were selected from the Semantic3D benchmark, as shown in Fig. 2.

2.2.2. Experimental data 1: Holyrood–Paradise dataset

This experimental dataset in Fig. 3 was captured by a Velodyne LiDAR sensor attached to a DJI M600 drone over two selected areas, Paradise and Holyrood, in Newfoundland,

Fig. 9. Dataset 6 of visual LiDAR odometry and mapping pipeline data captured by Bell412.



Canada. The data collected in Paradise, which includes a football ground surrounded by vegetation, are defined as the Paradise dataset. The data captured in Holyrood, which consists of a marine base parking lot, are defined as the Holyrood dataset. Unlike the benchmark dataset classes, we are interested in two classes, i.e., landable and non-landable, where non-landable regions can also include flat surfaces like water bodies, frozen lakes, marshlands, unstable ground, etc. Table 1 shows the number of points in each dataset and the number of points in the downsampled sets that are used to test the transfer-learned NN model. The objective of using the downsampled set in testing is to experiment with whether the NN model can perform at a reasonable speed with a VLOAM pipeline.

2.2.3. Experimental data 2: post-processed Holyrood dataset

The proposed architecture described in Section 2.1 can generate a local map with approximately 10 000 points at a time. To test the runtime-accuracy trade-off of the selected models, the downsampled Holyrood test set is divided into several sub-point clouds with varying numbers of points. As shown in Fig. 4, the full point cloud is segmented into 11 sub-clouds using CloudCompare software. Table 2 shows the number of points in each segmented point cloud.

2.2.4. Experimental data 3: Lighthouse dataset

The third set of experimental data contains the pipeline data, which are the local maps generated by the VLOAM module of the proposed system. Figure 5*a* shows the point cloud map generated by the VLOAM subsystem, which was implemented on a drone, and Fig. 5*b* shows one of the images captured by the monocular camera mounted onto the same drone. For reference purposes, this dataset will be called the Lighthouse dataset throughout the manuscript.

2.2.5. Experimental data 4: Bell412 dataset

This fourth experimental dataset was captured using a custom payload mounted on a Bell412 helicopter-generating point clouds using a VLOAM architecture. The payload design is shown in Fig. 6*a*, and the Bell412 helicopter used for taking the data is shown in Fig. 6*b*. For reference, this dataset will be termed the Bell412 dataset.

Figure 7 illustrates the path of the Bell412 helicopter on a Google Satellite map based on the GPS data taken by the payload set-up. Figures 8 and 9 show two sample point cloud maps generated by the VLOAM pipeline from the point clouds captured by the custom payload of the Bell412 helicopter. The colors in these two images of the point clouds are generated by the CloudCompare software based on intensity values for visualization.

Table 3 shows the number of points in the two datasets captured by the Bell412 helicopter and the approximate dimensions of the bounding box of each generated point cloud.

2.3. Labelling

The process flow diagram of the LZ labeling process of the Semantic3D dataset and the experimental data 1 is shown in Fig. 10. The method used is from Scherer et al. (2012), which uses a patch-based processing pipeline using xyz data of a point cloud. Since the classical method is carried out on structured point clouds, the point clouds are ordered depending on the x and y positions of each point. The point cloud is divided into 3 m $\times$ 3 m patches, considering the dimensions of the DJIM600 quadcopter and ± 1 m navigation accuracy. Then, planes are fitted to all patches. LZs are identified as the patches having the standard deviation of the z-axis less than 0.5 m and maximum z-deviation less than 6 m. Besides, the number of points per patch should exceed 15 points to be considered an LZ, ensuring an average point density of 1 point at every 20 cm $\times$ 20 cm region for a 3 m $\times$ 3 m patch (Scherer et al. 2012). A patch is chosen as landable if its slope is below 5°.

The algorithm in Fig. 10 could not detect water bodies, and it wrongly classified water areas as landable zones, as shown in Fig. 11. This was resolved using a semantic labeling app by manually selecting water areas and naming the point label as non-landable.

2.4. Model selection

As described in the introduction section, the initial work of point-based point cloud semantic segmentation can be found in (Qi et al. 2017*a*) and (Qi et al. 2017*b*). Due to the ineffectiveness of those two models in terms of runtime-accuracy trade-off and detecting several object classes, ConvPoint model (Boulch 2020) was selected based on the Semantic3D (Hackel et al. 2017) leaderboard and the online, operational requirement. In the ConvPoint model, a continuous convolutional layer was designed by adopting the discrete convolution used for images. As shown in Fig. 12, the segmentation network

Table 3. Number of points in the two Bell412 datasets and the dimensions of the bounding box of the datasets.

Dataset	No. of points in the dataset	Dimensions in metres (x, y, and z)
Bell412 Dataset 1	15 377 432	1448.21, 398.78, and 113.208
Bell412 Dataset 6	21 230 524	1042.34, 1036.17, and 143.125

Fig. 10. Process flow of the geometric labeling. LZ, landing zone.

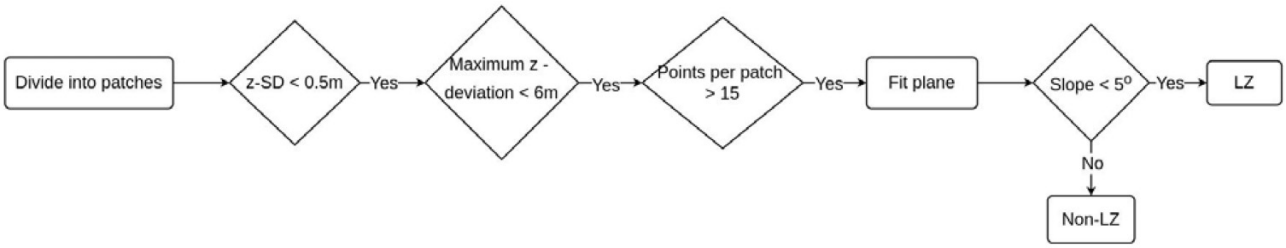
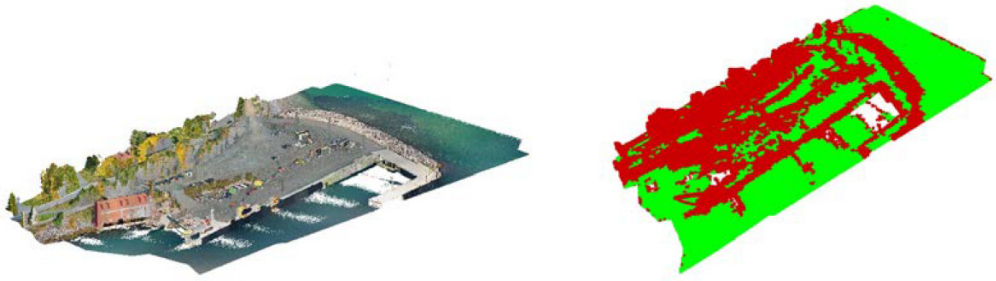


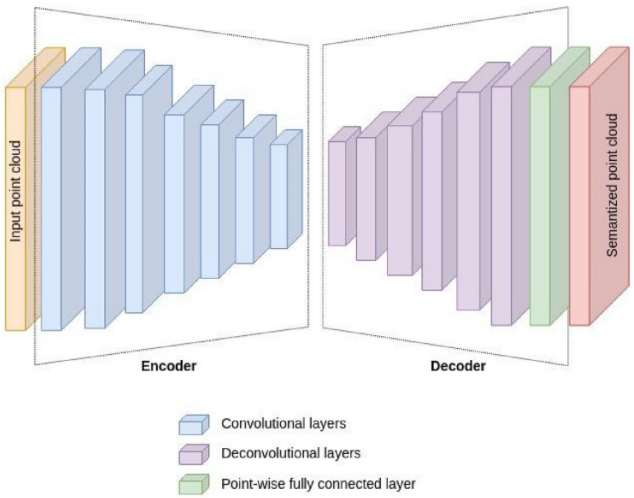
Fig. 11. Original Holyrood dataset and the colored landing zones (LZs) generated by geometric labeling. RGB, red, green, and blue.



(a) RGB colored Holyrood dataset without outliers

(b) Colored LZs in the Holyrood dataset (Green:landable, Red:non-landable)

Fig. 12. Sample illustration of ConvPoint semantic segmentation model.



has an encoder–decoder structure, where the encoder is a stack of convolutions that reduces the cardinality of the point cloud, while the decoder contains a stack of convolutions with skip connections and a total of 12 convolutional layers. The last layer is a point-wise linear layer used to generate an output dimension corresponding to the number of classes. At training time, the model randomly selects points in the considered point cloud and extracts all the points in an infinite 8 m vertical column centered on this selected point. During testing, the model computes a 2 D occupancy pixel map with a "pixel" size of 0.5 m outdoor scenes by projecting vertically on the horizontal plane. Then, the model considers each occupied cell as a center for a column (the same size as for training). For each column, it randomly selects a given number of points (i.e., sampling size) that are fed as input to the network. Finally, the output scores are aggregated at the point level, and points not seen by the network will be given the label of their nearest neighbor.

2.5. Transfer learning

Transfer learning is a deep learning approach where a model developed and trained on one task (base data) is reused as the starting point of a second task (target data). The pre-trained model weights will be fine-tuned to the new data to achieve significantly higher performance, and this is only suitable if the target data are similar to the base data.

For transfer learning the ConvPoint model for LZ identification, we modified the source code to output two object classes

Fig. 13. Transfer-learning graphs.

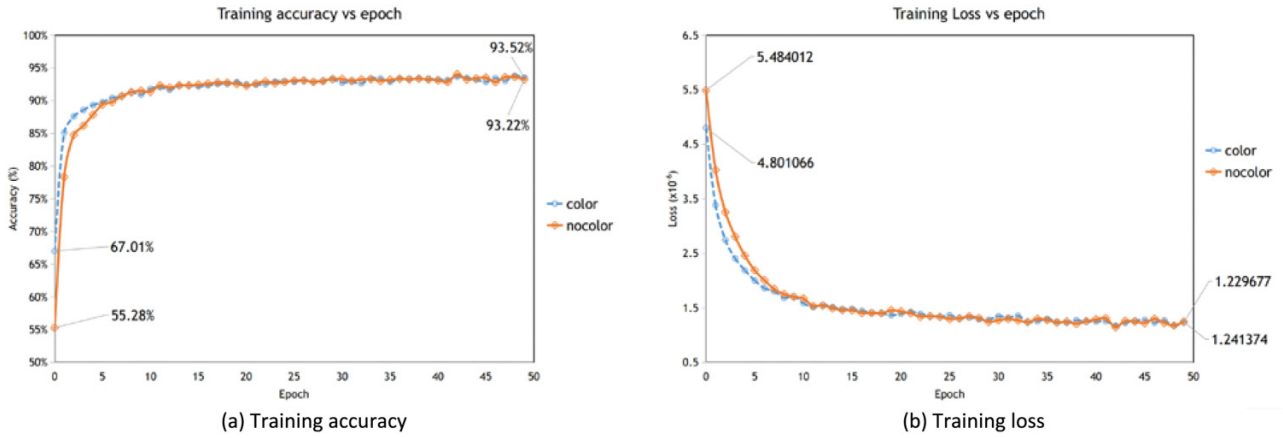
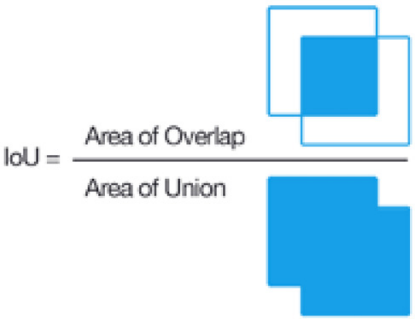


Fig. 14. Intersection over Union (IoU) calculation visualization. Source: Wikipedia.



(i.e., landable and non-landable). In evaluating the ConvPoint model, we used four transfer-learning modes derived based on the training sets used and the pretrained weights. For mode 1, we used the pretrained weights (pretrained on the Semantic3D dataset) as the initial checkpoint. We obtained these pretrained weights from the ConvPoint GitHub repository, which is available to the public. As the training set of the transfer learning of ConvPoint, we used the down-sampled Semantic3D dataset and a part of the Holyrood dataset containing points of the water bodies. The inputs contained 3 D point data, original RGB data, and landable/non-landable labels. The remaining subset of the Holyrood dataset and the Paradise dataset were used as the testing data. To analyze the effect of the use of RGB values, fine-tuning is done with and without the RGB data input. Figures 13*a* and 13*b* illustrate the transfer-learning accuracy and loss, respectively, for method 1, which indicates that having the low-resolution color data in the point cloud provides only insignificant performance gains. Therefore a separate NN model dealing with high-resolution color data is preferable to utilize color information.

2.6. Performance metrics

To evaluate the performance of different point cloud semantic segmentation models, we use mIoU, accuracy, average runtime, and throughput as the metrics. This section will briefly overview each metric.

2.6.1. Intersection over Union

Intersection over Union, commonly abbreviated as IoU, is another widely used metric in evaluating deep learning models for semantic segmentation. IoU can be defined as the area of overlap between the predicted segmentation and the ground truth divided by the area of union between the predicted segmentation and the ground truth. Figure 14 shows a generic visualization of IoU, while eq. 1 is the mathematical computation of IoU. The mIoU is the calculation of the IoU of each class and averaging the values.

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{1}$$

2.6.2. Accuracy

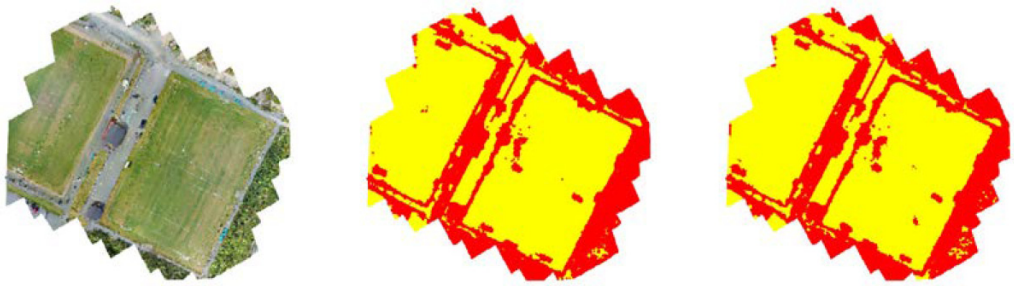
Informally, accuracy is the fraction of correct predictions over the total number of predictions assessed by a model. Equation 2 is the formal expression of the metric. Though accuracy can give an overview of how a model would perform, it should not be the only metric to be used when evaluating the performance of a model. When a model delivers high accuracy, it does not imply that the model has outperformed. The reason is most data have a class imbalance, where some object classes can dominate the point cloud while some make only a smaller portion. This cannot be ignored in a real-world navigation setting.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{2}$$

2.6.3. Average runtime

The average runtime speed is the average measure of time taken for the model to compute inference over a set of point

Fig. 15. Visualization of landing zone (LZ) detection for Paradise dataset—LZs are in yellow color and non-LZs are in red color.



(a) Paradise dataset (b) Predictions using Google Colab (c) Predictions using Jetson AGX

Fig. 16. Visualization of Landing zone (LZ) detection for Holyrood test dataset—LZs are in yellow color and non-LZs are in red color.

(a) Holyrood test dataset (b) Predictions using Google Colab (c) Predictions using Jetson AGX

Table 4. Nvidia Jetson AGX developer kit specifications.

	Technical specifications
GPU	512-Core Volta GPU with Tensor Cores
CPU	8-Core ARM v8.2 64-Bit CPU, 8 MB L2 + 4 MB L3
Memory	32 GB 256-Bit LPDDR4x | 137 GB/s

GPU, graphics processing unit.

cloud scans. We computed the average runtime speed as shown in eq. 3

$$\text{Average runtime} = \frac{\sum_{n=1}^{k} t_n}{k} \quad (3)$$

where
k = total number of scans
t_n = inference time of nth scan.

2.6.4. Throughput

As our objective is to find the most suitable real-time semantic segmentation model for point clouds, it is essential to identify how many points can be processed in a given time. For this, we computed the number of points processed during a unit time of seconds, which is known as throughput. This is computed using eq. 4,

$$\text{Throughput of } n\text{th scan} = \frac{\text{Total number of points in } n\text{th scan}}{\text{Inference time of } n\text{th scan}} \quad (4)$$

and averaged over the total throughput values.

3. Results

This section provides the main results of the point-based network evaluated on three variations of the datasets captured using the DJI M600. First, the results of the experimental data are presented, then the results of the post-processed experimental data will be presented, and finally, the results of newly acquired payload data will be described.

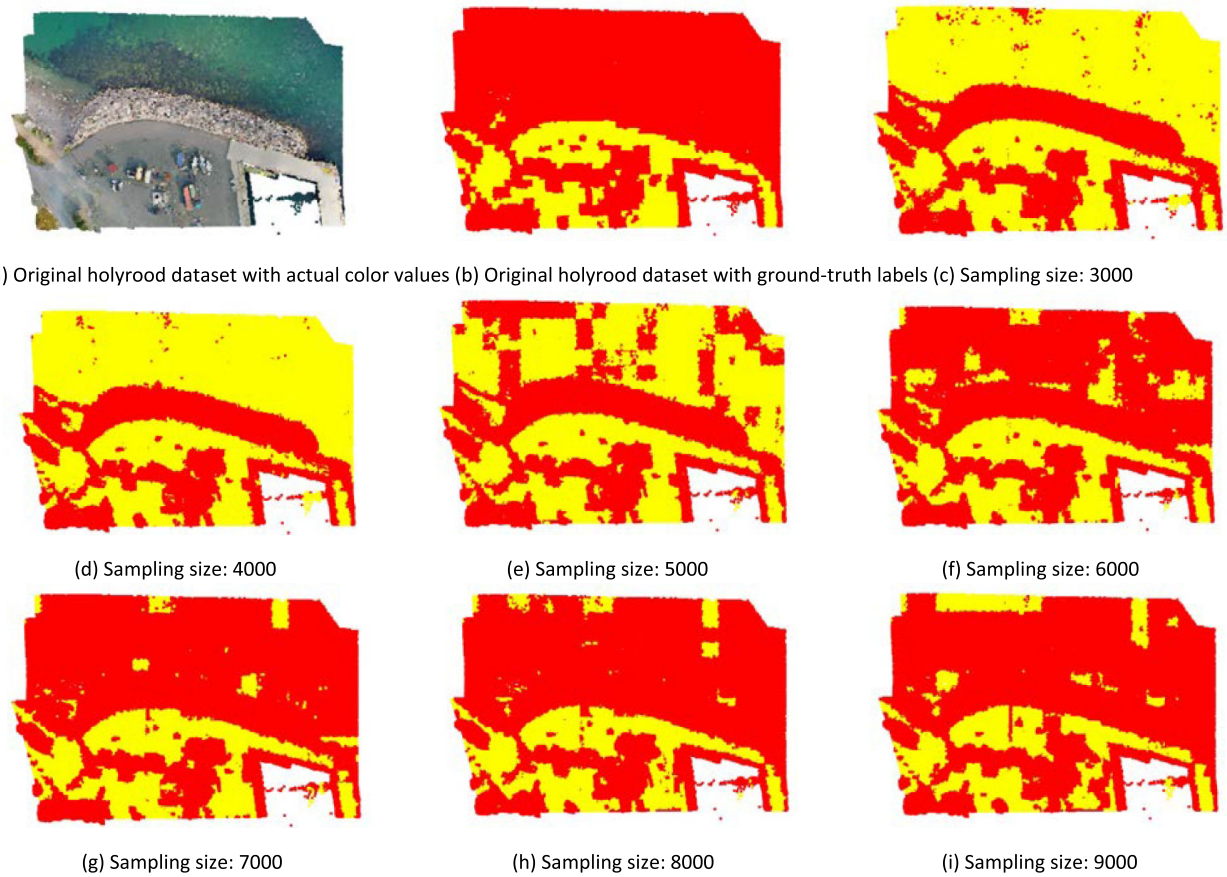
Nvidia Jetson AGX Xavier (Jetson AGX) developer kit was used to perform a comparative performance evaluation in the accuracy-runtime trade-off of the ConvPoint model for LZ detection for the different forms of the datasets. The Nvidia Jetson AGX is developed especially for autonomous machines, where it accelerates compute density and energy efficiency. Artificial Intelligence (AI) inference capabilities in intelligent machines like robots, factory systems, and UAVs can make use of this computing hardware given the 105 mm × 105 mm × 65 mm form factor, 630 g weight, and max 35 W power consumption. The main hardware specifications of Jetson AGX are listed in Table 4.

Table 5. Performance of ConvPoint model on the experimental datasets.

Dataset	Google Colab[a]			Jetson AGX Xavier[b]				
	Inference time (s)	Accuracy (%)	Throughput (pts/s)	Inference time (s)		throughput (pts/s)		Accuracy (%)
				15 W	MAXN	15 W	MAXN	
Paradise	57.83	89.72	3940.15	31.25	18.92	7291.11	12044.84	89.60
Holyrood	41.63	92.13	3633.85	29.16	14.78	5187.30	10238.37	91.68

[a]Google Colaboratory space equipped with Tesla T4 GPU
[b]The Jetson AGX module has several power modes in which it can be operated. These power modes differentiate the use of GPU resources. While 15W is the default power mode, MAXN mode utilizes all the available resources. GPU, graphics processing unit.

Fig. 17. Visualization of landing zone (LZ) detection for post-processed Holyrood merged point cloud—LZs are in yellow color and non-LZs are in red color.



3.1 Results: Experimental datasets

Figs. 15 and 16, and Table 5 report the quantitative and qualitative performance of the ConvPoint model for LZ detection on the original down-sampled experimental datasets. It is shown that ConvPoint fine-tuned model delivered comparable performance, especially in identifying the water regions, i.e., the Holyrood dataset. The fine-tuned ConvPoint model is evaluated on the Jetson AGX for prediction accuracy and inference time when two power modes are available. As summarized in Table 5, by using the maximum power mode, it has taken lesser inference time and generated higher throughput. However, the inference time of the ConvPoint model is comparatively long though it has scored a higher accuracy, which is a reasonable drawback for real-time point cloud semantic segmentation.

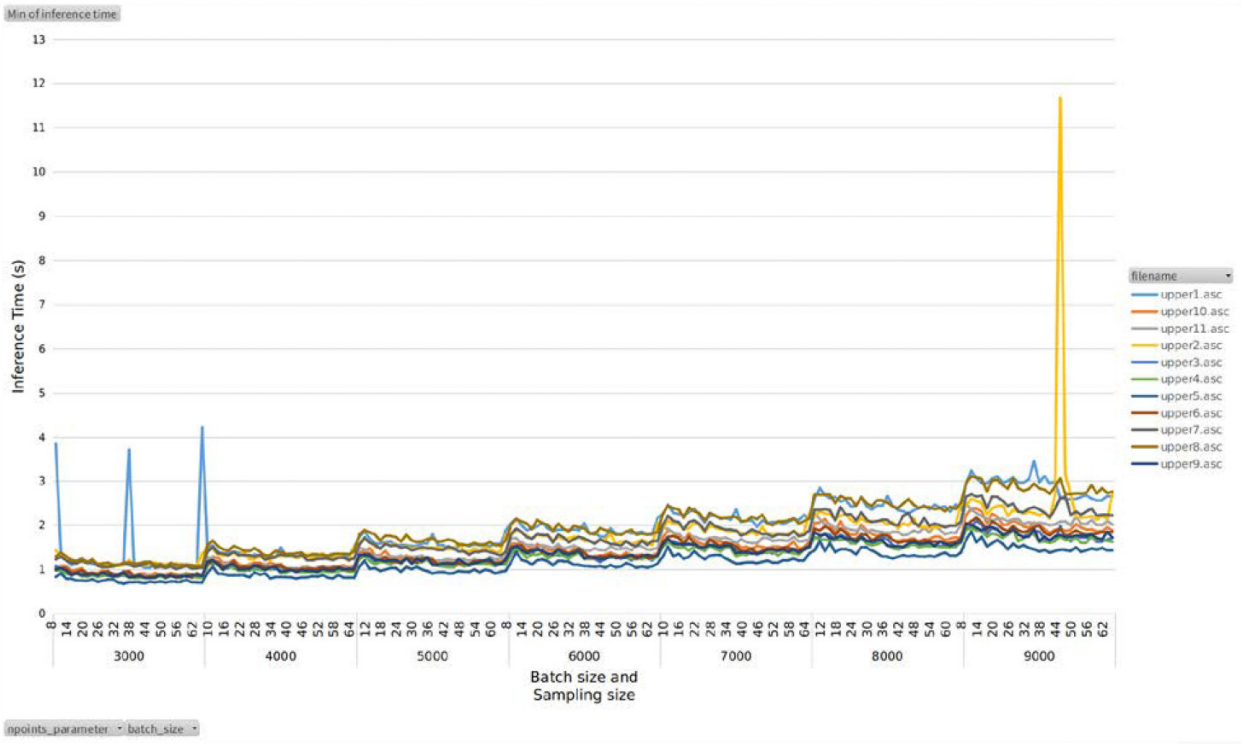
3.2. Results: post-processed experimental dataset

This section will illustrate the performance of the ConvPoint model on the sectioned experimental dataset described in Section 2.2.3 in the methods of qualitative, quantitative, and graphical evaluations.

Table 6. Performance of ConvPoint model on the post-processed experimental dataset for the sampling sizes 3000, 6000, and 9000.

	Inference times (s)			Throughput (pts/s)			Accuracy		
Sub-point cloud	3000	6000	9000	3000	6000	9000	3000	6000	9000
Sub-cloud 1	1.09	1.87	3.01	8318.40	4859.32	3016.23	22.23%	94.42%	88.94%
Sub-cloud 2	1.10	1.66	2.21	14 392.31	9532.94	7135.31	52.88%	95.85%	92.13%
Sub-cloud 3	0.87	1.33	1.80	13 294.26	8767.27	6453.84	44.46%	94.11%	97.98%
Sub-cloud 4	0.83	1.31	1.78	12 034.65	7585.73	5581.70	8.98%	72.12%	91.94%
Sub-cloud 5	0.72	1.09	1.46	12 513.06	8223.87	6167.99	0.70%	69.57%	99.99%
Sub-cloud 6	0.88	1.34	1.87	18 104.54	11 908.23	8540.46	88.99%	89.68%	89.00%
Sub-cloud 7	1.08	1.60	2.43	18 892.94	12 759.38	8394.01	93.02%	92.10%	91.91%
Sub-cloud 8	1.12	1.87	2.83	13 761.73	8228.74	5420.82	88.21%	87.95%	83.79%
Sub-cloud 9	0.87	1.34	1.83	17 641.84	11 390.12	8392.60	94.24%	93.67%	86.62%
Sub-cloud 10	0.92	1.42	2.02	16 581.09	10 704.60	7533.99	92.20%	94.85%	93.30%
Sub-cloud 11	0.89	1.48	2.09	15 323.76	9210.53	6506.59	70.65%	93.40%	95.18%

Fig. 18. Inference time verus batch size and sampling size for each partitioned dataset.



3.2.1. Qualitative evaluation

Figure 17 visualizes the LZs predictions computed by the ConvPoint model on the sectioned experimental dataset for different sampling sizes and a defined batch size of 24. In the visualization, the predictions on each sub-point cloud are displayed in a merged point cloud to demonstrate the qualitative accuracy of the output. Figure 17*a* is the original Holyrood test set with the actual color values, while Fig. 17*b* contains the color-coded LZs for the original dataset. Figures 17*c*–*i* show how the predictions change when the sampling size changes with a given batch size, i.e., 24.

3.2.2. Quantitative evaluation

Table 6 tabulates the quantitative performance of the ConvPoint model for the partitioned Holyrood test set for three different sampling sizes (i.e., 3000, 6000, and 9000) with a fixed batch size of 24 on Jetson AGX Xavier when Max power mode was set.

Fig. 19. Throughput versus batch size and sampling size for each partitioned dataset.

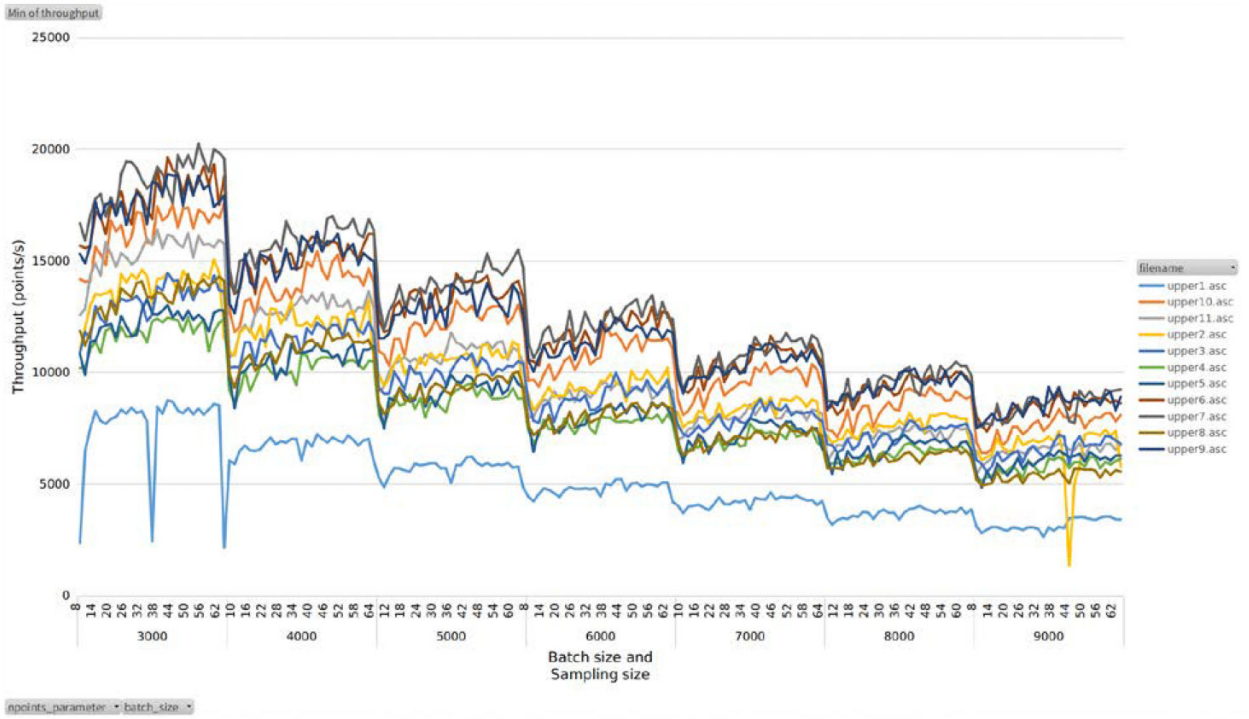


Fig. 20. Accuracy versus batch size and sampling size for each partitioned dataset.

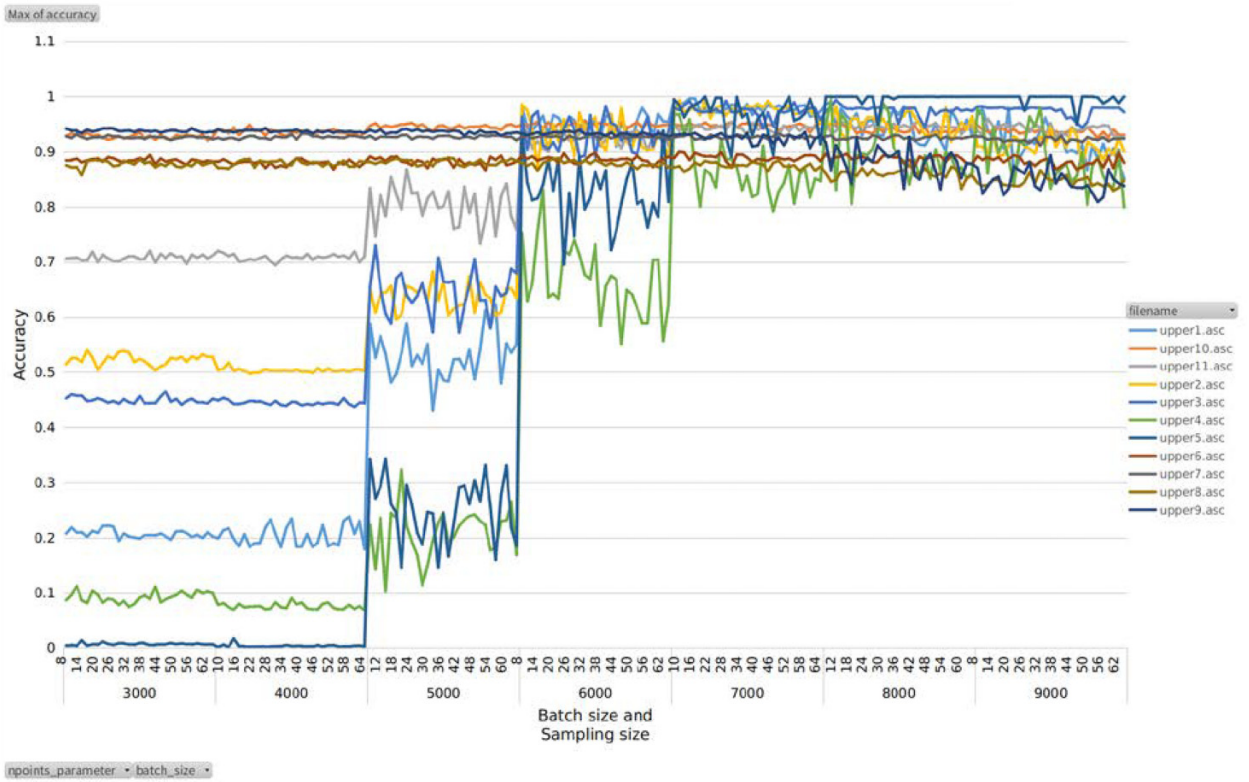


Fig. 21. Accuracy versus batch size and sampling size for each partitioned set containing water bodies.

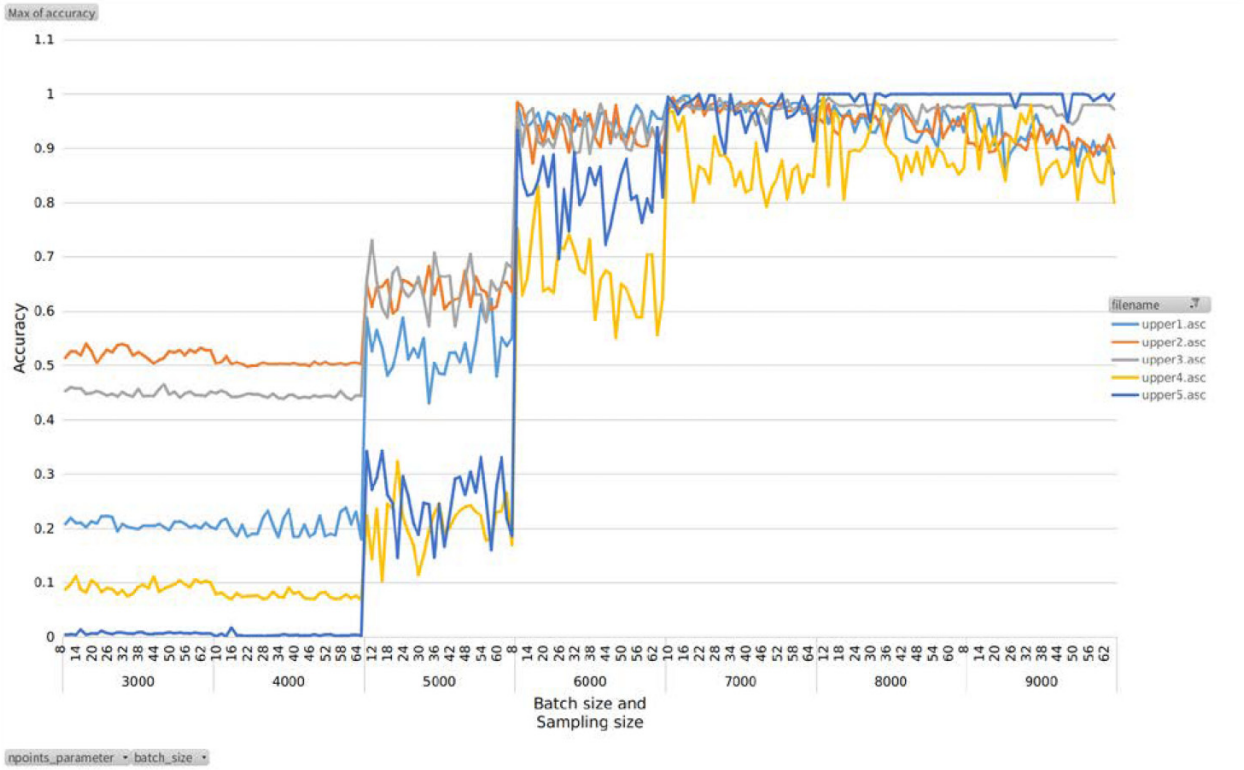
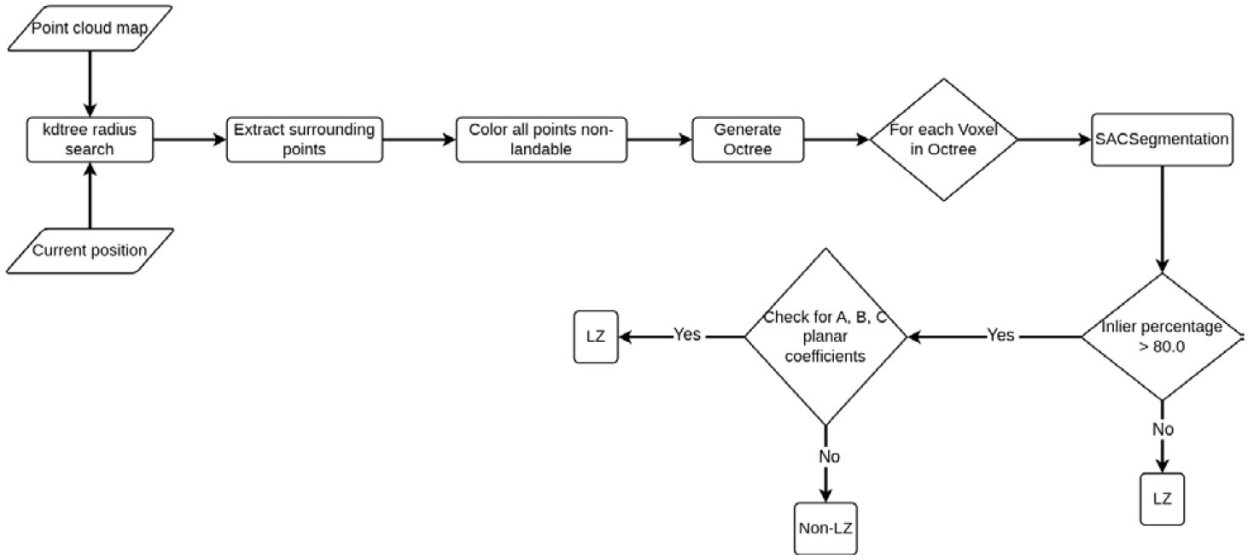


Fig. 22. Process flow of the labeling of pipeline data. LZ, landing zone.



3.2.3. Graphical evaluation

Figures 18–21 illustrate how the ConvPoint model performed when different combinations of batch sizes and sampling sizes are defined. Figure 21 shows how the accuracy would fluctuate for different batch sizes and sampling sizes on the partitioned sets containing water bodies.

Illustrations in sections 3.2.1–3.2.3 present how the performance of the ConvPoint model on the partitioned Holyrood test set would transform when different hyperparameters (i.e., batch size, sampling size) are selected. From Figs. 19 and 20, it can be perceived that a lower sampling size can generate high throughput. Although lower sampling sizes have trouble with water bodies detection, it is not the concern of our task as it will be handled by the image segmentation module. Therefore, a comparatively lower sampling size with a higher batch size and an inference time of 1 Hz is a

Fig. 23. Dataset 1 of visual LiDAR odometry and mapping pipeline data captured by Bell412.

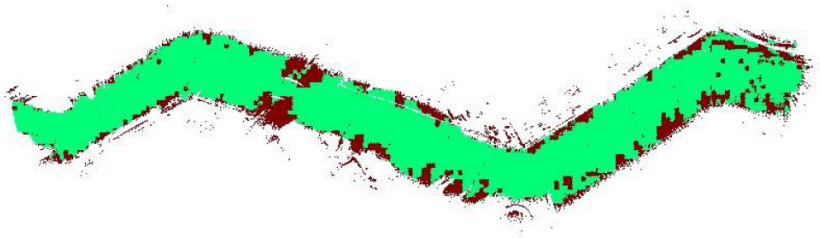


Fig. 24. Dataset 6 of visual LiDAR odometry and mapping pipeline data captured by Bell412.

suitable hyperparameter setting to gain a better trade-off for the VLOAM pipeline data rate.

4. Implementing the LZ detection module in the proposed pipeline

This section will explore the use of point cloud data presented in Section 2.2.5 captured using the VLOAM pipeline described in Section 2.1.

4.1. Labeling pipeline data

Since no ground truth labels are available for the pipeline data in Section 2.2.5, it is necessary to label the data for landable and non-landable zones. The labeling process in Fig. 10 cannot be used for the pipeline data as it works on generated maps and is written in Python, which can take a long time to generate labels. Therefore, an efficient VLOAM-compatible algorithm was designed using C++ programming, with the capability to integrate into the VLOAM subsystem. The flow diagram in Fig. 22 shows the algorithm developed to achieve this objective. This method can render the LZ labels for each point cloud, while generating the point cloud map and coloring the map accordingly.

Figures 23 and 24 illustrate the LZs generated by the algorithm in Fig. 22 for two datasets captured by the payload on a Bell412 helicopter.

4.2. Class balancing method

As seen in Figs. 23 and 24, the ground truth labels of VLOAM data do not have a uniform class balance between LZs and non-LZs. To address this problem, we used class balancing to generate trainable data from the VLOAM data. First, we generated $100 \times 100 \times z$ blocks from the Bell412 data using CloudCompare software. Then we exported the blocks with both landable and non-landable zones and had more than 10 data points. We programmed a Python script to check each exported block for a class balance between 60% and 40% and export the blocks within that range as the final training set.

4.3. Transfer-learning methods

Table 7 shows the training sets and the pretrained weights used in four transfer-learning modes. The first transfer-learning mode, Mode 1, is similar to that discussed in Section 2.5. For modes 2, 3, and 4, we included the class-balanced Bell412 data segments to evaluate the use of VLOAM data for LZ determination. We used different combinations of training data to derive the best transfer-learning approach for VLOAM data that can produce higher accuracy and generalization.

4.4. Experimental results

This section presents the predictions generated by each transfer-learning method on test datasets. Figures 25–28 show how the trained ConvPoint model from each transfer-learning mode has predicted the LZs on different test sets. We evaluated different types of test point clouds captured using different architectures for LZ predictions in each mode to assess the generalization of the training approach.

As illustrated in Fig. 27, LZ predictions produced by mode 4 on the Paradise dataset have a significant difference with the ground truth labels compared to other modes. Therefore, we dismiss mode 4 as a candidate for transfer learning on VLOAM data. As illustrated in Figs. 25–28, the LZ predictions output by modes 2 and 3 are nearly similar, with marginal improvement in the predictions of mode 3 visible in the predictions on the Holyrood dataset.

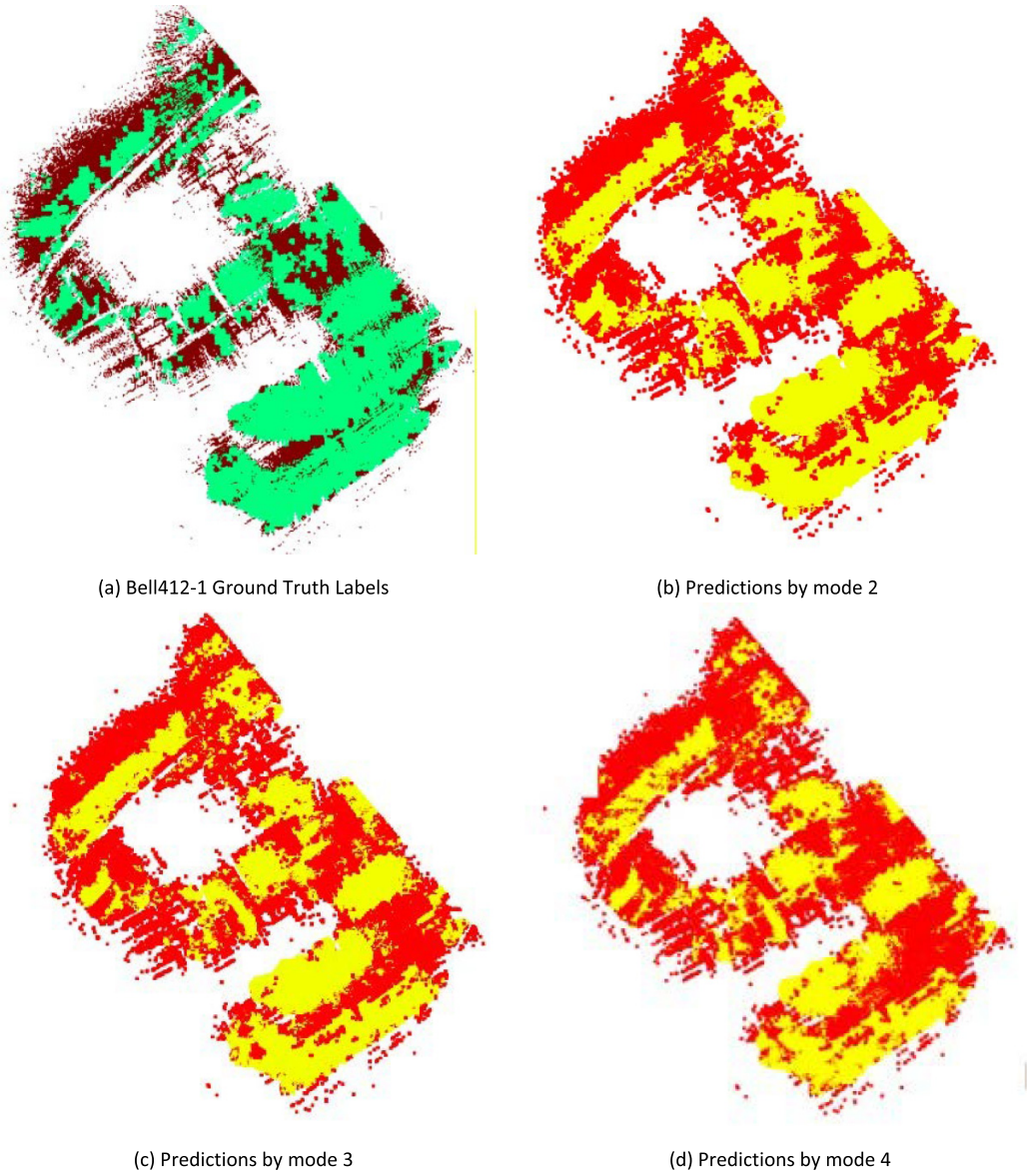
The results of the Bell412 datasets (Figs. 25 and 26) demonstrate the ability of the proposed point-based LZ evaluation

Table 7. Transfer-learning modes based on the training set and pretrained weights.

Transfer learning Mode	Training set	Pretrained weights
Mode 1	Semantic3D LZs + Holyrood LZs	Original Semantic3D
Mode 2	Class balanced Bell412 data segments	Original Semantic3D
Mode 3	Class balanced Bell412 data segments	Original LZ trained (Section 2.5)
Mode 4	Class balanced Bell412 data segments + Semantic3D LZs + Holyrood LZs	Original Semantic3D

Note: LZ, landing zone.

Fig. 25. Predictions on Bell412-1 by each transfer-learning mode.

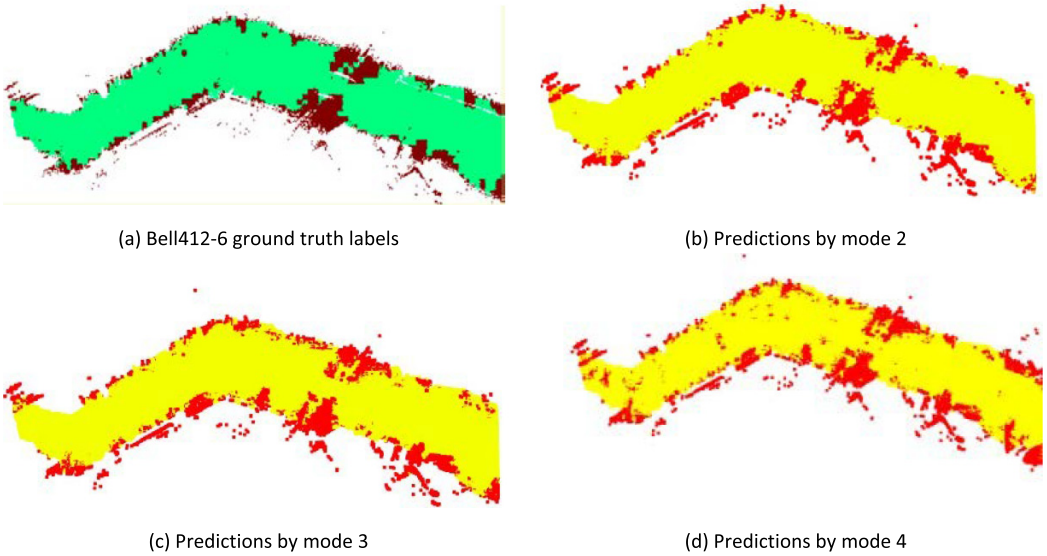


module to operate within a VLOAM navigation pipeline to generate maps at a target update rate (≈1 Hz). It should be noted that modes 2 and 3 did not deliver good predictions for the water body portion of the Holyrood test set as in Fig. 28. We can address this issue by fusing image segmentation and object detection labels with the LZ labels introduced in Section 1. The addition of the image modules will be investigated as part of future developments of this work.

5. Conclusion

This paper proposed a novel point-based NN LZ detection architecture that can operate with a VLOAM navigation pipeline and investigates the accuracy-runtime trade-offs of the method for real-time applications. Based on the Semantic3D benchmark leaderboard, ConvPoint architecture was selected as the target model for the task. The work investigated different combinations of hyperparameters, i.e., batch

Fig. 26. Predictions on Bell412-6 by each transfer-learning mode.



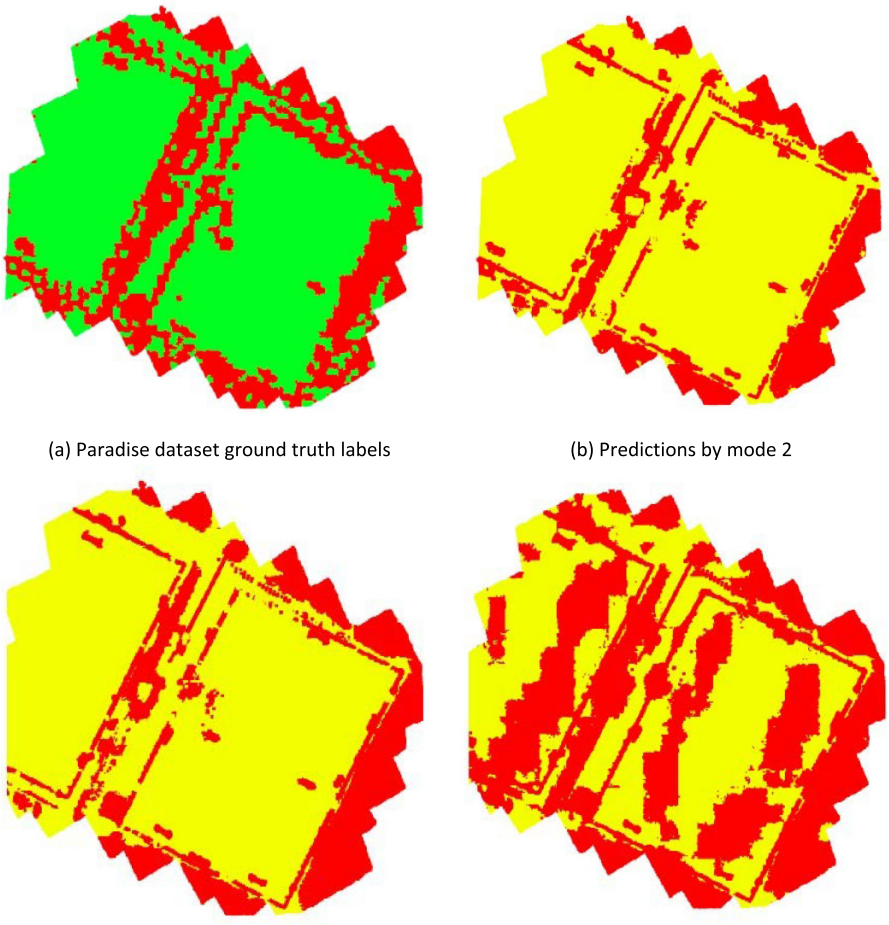
(a) Bell412-6 ground truth labels

(b) Predictions by mode 2

(c) Predictions by mode 3

(d) Predictions by mode 4

Fig. 27. Predictions on Paradise dataset by each transfer-learning mode.



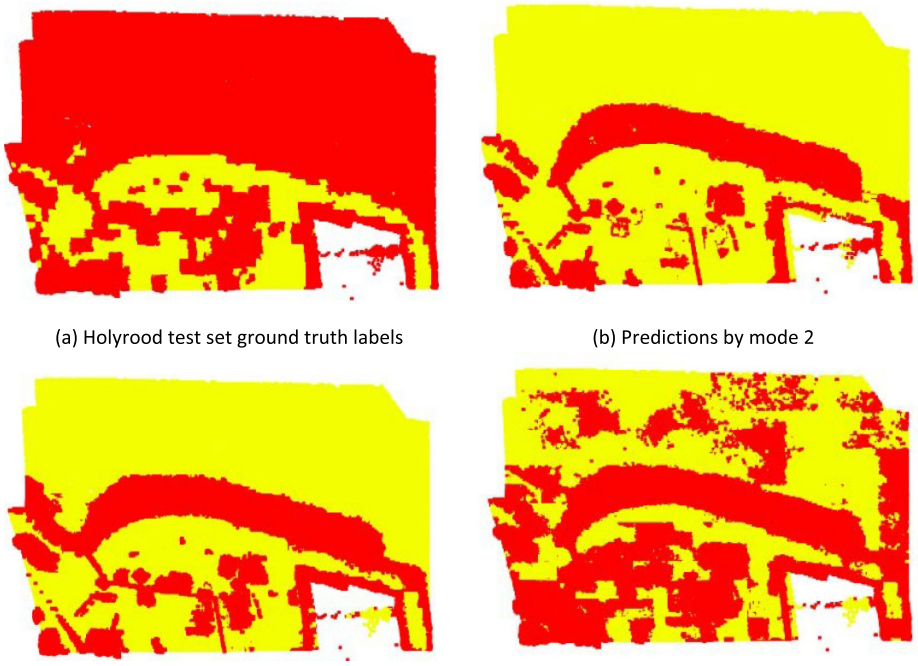
(a) Paradise dataset ground truth labels

(b) Predictions by mode 2

(c) Predictions by mode 3

(d) Predictions by mode 4

Fig. 28. Predictions on the Holyrood test set by each transfer-learning mode.



size and sampling size, in terms of the performance metrics, i.e., inference time, throughput, and accuracy. Validation of the method was performed using custom datasets captured on a DJI M600 drone and a Bell412 aircraft to generate the LZ module's maps at a target update rate (≈1 Hz), while operating within a VLOAM navigation pipeline. Accurate detection of water bodies, marshlands, and low vegetation as non-landable is crucial for VTOL operations. From the results described in this paper, it is evident that to get a comparatively accurate detection of water areas in the given dataset, a larger sampling size should be set, which also can lead to lower throughput (higher inference time). This bottleneck can be resolved by fusing the semantic labels generated by the point cloud segmentation with the pixel labels generated by the color image semantic segmentation of the same region and by using a broader range of datasets to train the NN model. The fusion of the point cloud labels, pixel labels, and object tracks to deliver combined labels with bounding boxes is the next step of the proposed architecture. Additional transfer learning on more data is also required to detect the full region of LZs like helipads and will be achieved in the next steps of this task. Additionally, future work will include TensorRT optimizations, further simplification of the NN model, and C++ programming optimization on GPU utilization to achieve improved update rates of the LZ detection module.

Acknowledgements

The authors would like to acknowledge Ms. Sachithra Athapattu for her contribution to the LZ labeling algorithm.

Article information

History dates
Received: 26 July 2022
Accepted: 12 October 2023
Accepted manuscript online: 12 February 2024
Version of record online: 2 April 2024

Data availability
Data generated during this study are available from the corresponding author upon reasonable request.

Author information

Author ORCIDs
Narmada M. Balasooriya https://orcid.org/0000-0002-0651-6572

Author notes
Awantha Jayasiri served as an Associate Editor at the time of manuscript review and acceptance; peer review and editorial decisions regarding this manuscript were handled by another Editorial Board Member.

Author contributions
Conceptualization: NMB, ODS

Data curation: ODS
Formal analysis: NMB
Funding acquisition: ODS, AJ, GM
Investigation: ODS
Methodology: NMB
Project administration: ODS, AJ, GM
Resources: ODS, AJ, GM
Software: NMB
Supervision: ODS, AJ, GM
Validation: NMB
Visualization: NMB
Writing – original draft: NMB
Writing – review & editing: ODS

Competing interests

The authors declare there are no competing interests.

Funding information

This research was supported by National Research Council, Canada.

References

Boulch, A. 2020. Convpoint: continuous convolutions for point cloud processing. Comput. Graph. **88**: 24–34.

Desaraju, V.R., Michael, N., Humenberger, M., Brockers, R., Weiss, S., Nash, J., and Matthies, L. 2015. Vision-based landing site evaluation and informed optimal trajectory generation toward autonomous rooftop landing. Auton. Robots, **39**(3): 445–463. doi:10.1007/S10514015-9456-X.

Garg, M., Kumar, A., and Sujit, P.B. 2015. Terrain-based landing site selection and path planning for fixedwing UAVs. 2015 International Conference on Unmanned Aircraft Systems, ICUAS: Denver, CO, USA, **2015**. pp. 246–251. doi:10.1109/ICUAS.2015.7152297.

Hackel, T., Savinov, N., Ladicky, L., Wegner, J.D., Schindler, K., and Pollefeys, M. 2017. SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci. *IV-1-W1*, 91–98.

Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., et al. 2022. Learning semantic segmentation of large-scale point clouds with random sampling. IEEE Trans. Pattern Anal. Mach. Intell. **44**: 8338–8354. doi:10.1109/TPAMI.2021.3083288.

Kaljahi, M.A., Shivakumara, P., Idris, M.Y.I., Anisi, M.H., Lu, T., Blumenstein, M., and Noor, N.M. 2019. An automatic zone detection system for safe landing of UAVs. Expert Syst. Appl. **122**: 319–333. doi:10.1016/J.ESWA.2019.01.024.

Lee, M.F.R., Nugroho, A., Le, T.T., Bahrudin, and Bastida, S.N. 2020. Landing area recognition using deep learning for unmanned aerial vehicles. International Conference on Advanced Robotics and Intelligent Systems, ARIS, Taipei, Taiwan. pp. 1–6. doi:10.1109/ARIS50834.2020.9205793.

Lee, S., and Kwon, Y. 2020. Safe landing of drone using AI-based obstacle avoidance. Int. J. Mech. Eng. Robot. Res. **9**(11). doi:10.18178/ijmerr.9.11.1495-1501.

Long, J., Shelhamer, E., and Darrell, T. 2015. Fully convolutional networks for semantic segmentation. Pro-ceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR). Boston, MA, 2015. pp. 3431–3440.

Lorenzo, O.G., Martínez, J., Vilariño, D.L., Pena, T.F., Cabaleiro, J.C., and Rivera, F.F. 2016. Landing sites detection using LiDAR data on manycore systems. J. Supercomput. **73**(1): 557–575. doi:10.1007/S11227-016-1912-7.

Lusk, P.C., Glaab, P.C., Glaab, L.J., and Beard, R.W. 2019. Safe2ditch: emergency landing for small unmanned aircraft systems. J. Aerosp. Inf. Syst. **16**(8): 327–339. doi:10.2514/1.I010706.

Maturana, D., and Scherer, S. 2015a. 3d convolutional neural networks for landing zone detection from lidar. 2015 IEEE international conference on robotics and automation (ICRA), Seattle, WA, 2015. pp. 3471–3478.

Maturana, D., and Scherer, S. 2015b. Voxnet: A 3d convolutional neural network for real-time object recognition. 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), Hamburg, Germany, 2015. pp. 922–928.

Milioto, A., Vizzo, I., Behley, J., and Stachniss, C. 2019. RangeNet++: fast and accurate LiDAR semantic segmentation. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).

Polvara, R., Sharma, S., Wan, J., Manning, A., and Sutton, R. 2019. Autonomous vehicular landings on the deck of an unmanned surface vehicle using deep reinforcement learning. Robotica, **37**(11): 1867–1882. doi:10.1017/S0263574719000316.

Qi, C.R., Su, H., Mo, K., and Guibas, L.J. 2017a. Pointnet: deep learning on point sets for 3d classification and segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition, Honolulu, HI. pp. 652–660.

Qi, C.R., Yi, L., Su, H., and Guibas, L.J. 2017b. Pointnet++: deep hierarchical feature learning on point sets in a metric space. Conference on Computer Vision and Pattern Recognition. *abs/1706.02413*. Available from http://arxiv.org/abs/1706.02413 [accessed November 2020].

Scherer, S., Chamberlain, L., and Singh, S. 2012. Autonomous landing at unprepared sites by a full-scale helicopter. Robot. Auton. Syst. **60**(12): 1545–1562. doi:10.1016/j.robot.2012.09.004.

Thomas, H., Qi, C.R., Deschaud, J.-E., Marcotegui, B., Goulette, F., and Guibas, L.J. 2019. Kpconv: flexible and deformable convolution for point clouds. *In* Proceedings of the IEEE International Conference on Computer Vision. Seoul, Korea (South). pp. 6410–6419.

Warren, M., Mejias, L., Yang, X., Arain, B., Gonzalez, F., and Upcroft, B. 2015. Enabling aircraft emergency landings using active visual site detection. Field Serv. Robot., 167–181. doi:10.1007/978-3-319-07488-7_12.

Wu, B., Wan, A., Yue, X., and Keutzer, K. 2018. Squeezeseg: convolutional neural nets with recurrent crf for real-time road-object segmentation from 3d lidar point cloud. 2018 IEEE International Conference on Robotics and Automation (ICRA). pp. 1887–1893.

Yan, L., Qi, J., Wang, M., Wu, C., and Xin, J. 2020. A safe landing site selection method of uavs based on lidar point clouds. 2020 39th Chinese Control Conference (CCC), Shenyang, China. pp. 6497–6502.

Ye, X., Li, J., Huang, H., Du, L., and Zhang, X. 2018. 3D recurrent neural networks with context fusion for point cloud semantic segmentation. Proceedings of the European Conference on Computer Vision (ECCV), Munich, Germany, 8–14 September 2018. pp. 403–417.

Zhang, J., and Singh, S. 2015. Visual-lidar odometry and mapping: low-drift, robust, and fast. 2015 IEEE International Conference on Robotics and Automation (ICRA), IEEE International Washington State Convention Center Seattle, Washington, 26–30 May 2015. pp. 2174–2181.