Development of Deep Learning Modules for Autonomous Navigation in Marine and Aerial Robotic Applications

by

©Narmada M. Balasooriya

A Thesis submitted to the School of Graduate Studies in partial fulfillment of the requirements for the degree of

M.Eng

Faculty of Engineering and Applied Science

Memorial University of Newfoundland

February 2023

St. John's Newfoundland

Abstract

This thesis develops two studies on deep learning-based autonomous navigation systems for marine and aerial field robotic applications.

The first study involves developing a sea ice detection module to support the autonomous navigation of icebreakers using image semantic segmentation. This module aims to distinguish sea ice from water, sky, and the ship's body when images captured onboard an icebreaker are received from a shipborne camera. The study compares the performance of the previous work on sea ice detection by the PSPNet model with a new-state-of-the art image semantic segmentation model called DeepLabv3. To evaluate the DeepLabv3 model, it is transfer-learned on the same image data used for the PSPNet model. The performance of both models is tested on a navigation module equipped with a Jetson AGX Xavier developer kit using standard evaluation metrics.

The second study contains the development of a landing zone detection pipeline using Lidar semantic segmentation to support the vertical take-off and landing vehicle autonomy. The study evaluates different point cloud semantic segmentation approaches for their compatibility with the landing zone detection task. The main objective of this study is to use only the Lidar data to detect safe landable zones using deep learning-based architectures and to achieve an accuracy-runtime trade-off for real-time operations. The performance of the neural network models for point cloud semantic segmentation is evaluated using standard metrics and different variations of aerial Lidar data. The study also assesses the feasibility of integrating the landing zone detection module into a visual Lidar odometry and mapping pipeline for faster inference by the neural network models.

Acknowledgements

I sincerely thank my supervisor, Dr. Oscar de Silva, for consistently guiding and tolerating all my mistakes and setbacks.

I wholeheartedly appreciate the support from the co-supervisors, Dr. George Mann and Dr. Awantha Jayasiri, for their advice which proved monumental towards the success of my study.

The physical and technical support from the Faculty of Engineering and Applied Science of the Memorial University of Newfoundland is immeasurable. Without their help, this thesis would not have been a reality.

I would acknowledge the constant love and appreciation from my parents, sister, and fiancée for holding onto me during hard times.

I wish to appreciate the support and motivation from my colleagues and friends, who stood by my side when I was in need. I would like to acknowledge Ms. Sachithra Athapattu for her contribution to the landing zones labeling algorithm.

Finally, I would like to appreciate the support of my colleague Mr. Kusal Tennakoon, in accomplishing the most complicated challenges encountered.

Your assistance and moral support during my study are highly recognized.

Table of Contents

List of Tables

List of Figures

Abbreviations

ABS	American Bureau of Shipping
AI4L	Artificial Intelligence for Logistics
API	Application Programming Interface
CNN	Convolutional Neural Network
COCO	Common Objects in COntext
DoF	Degrees of Freedom
FCN	Fully Convolutional Network
FOV	Field of view
GPS	Global positioning system
GPU	Graphical Processing Unit
HETC	Harsh Environment Technology Centre
IMU	Inertial measurement unit
IoU	Intersection over Union
ISL	Intelligent Systems Lab
JAX Kit	Nvidia Jetson AGX Xavier
kNN	k-Nearest Neighbour
KPConv	Kernel Point Convolution
Lidar	Light Detection And Ranging
LZ	Landing Zone
MAV	Micro-aerial vehicle
mIoU	Mean Intersection over Union

MLP	Multi-Layer Perceptron
MUN	Memorial University of Newfoundland
NN	Neural Network
NRC	National Research Council
PCA	Principal Component Analysis
PCL	Point Cloud Library
PROSAC	Progressive Sample Consensus
PSPNet101	Pyramid Scene Parsing Network
R-CNN	Region-based Convolutional Neural Network
RANSAC	Random Sample And Consensus
RGB	Red Green Blue
RNN	Recurrent Neural Network
ROS	Robot Operating System
SAR	Synthetic Aperture Radar
SLAM	Simultaneous Localization And Mapping
VILOAM	Visual Inertial Lidar Odometry and Mapping
VINS	Visual Inertial Navigation System
VLOAM	Visual Lidar Odometry and Mapping
VTOL	Vertical Take-Off and Landing Vehicle

Chapter 1

Introduction

Deep Learning is a subset of machine learning where the learning algorithms are inspired by the human brain's structure and functions and attempt to mimic the neural processing method of the brain [1]. Deep learning differs from machine learning in the context of the data complexity, feature detection algorithm, and performance. The backbone of deep learning is made of neural networks, and most deep learning algorithms have more than three layers in their neural networks [1], [2].

One of the main features of deep learning is its building structure, where these models can learn how simple parts form a complex entity. The hierarchical nature of the layer composition and the supervised learning approach achieves learning from small compositions to complex objects. Adding more layers into the network and more units within a layer enables a deep network to represent functions with higher complexity. When sufficiently large training data and models are given, deep learning methods can achieve human-like accuracy in mapping an input vector to an output vector. This mapping between input and output is accomplished by parametric function approximation, which is the primary building block of deep learning algorithms. This function approximation is computed by the deep feedforward networks [1], where the model learns the value of the parameters in a mapping function. For the model to get an accurate function approximation, it requires ample data with training labels [1].

Autonomous navigation helps a vehicle or a robotic system plan and execute its path based on the perceptions of the environment acquired through various sensor data. For better scene understanding, autonomous systems must perceive obstacles, objects, traffic signs, passengers, and animals. With the advances in end-to-end learning in deep neural networks, perception using images and Lidar sensors allows autonomous systems to navigate accurately [3]. With

the availability of faster *Graphical Processing Unit*s (GPUs) and large-scale labeled datasets [4], deep neural networks can understand different objects and scenarios in the environment without overfitting to the training data.

The intelligent systems lab at the Memorial University of Newfoundland conducts applied research to study vessels and offshore structures operating in harsh environments by collaborating with the *American Bureau of Shipping* (ABS) and the Harsh Environment Technology Center (HETC) at Memorial University. Under these projects, different deep learning modules are investigated for developing autonomous marine and aerial perception and navigation systems.

Online sea-ice detection is one of the projects that the intelligent systems lab collaborates with ABS HTEC. With the increasing number of shipping vessels operating in polar regions where sea-ice are abundant, accurate detection of them is essential for safe navigation. The sea-ice detection project focuses on what deep learning models can be deployed on the required hardware platforms requested by the navigators. The main reasons to consider deep learning models are their object detection, segmentation capability, and performance accuracy to support informed decision-making and the future marine autonomy of the platforms.

The intelligent systems lab is conducting industrial collaborations with the flight research lab of the *National Research Council* (NRC) on integrating artificial intelligence systems into flight planning. One of the modules under the artificial intelligence for flight control project of the intelligent systems lab and the NRC is detecting safe landable zones for vertical take-off and landing vehicles. Accurate detection of a safe landing zone is crucial when an aerial vehicle has to maneuver an unplanned landing. Automating the landing-zone detection will help the pilot's visuals and support the aircraft's autonomous landing capabilities.

The main objective of this thesis is to evaluate the performance of deep learning algorithms for the two independent applications discussed above, which have varying environmental conditions. In this chapter, the motivations, problem definitions, objectives, and contributions for each study are presented, followed by the organization of the thesis.

1.1 Sea-ice detection using image semantic segmentation

1.1.1 Motivation

When seawater is frozen and floats on the ocean surface, that is commonly known as sea ice. Sea ice plays a major role in polar ecosystems, global climate, and ship navigation [5]. This study focuses on sea ice detection for the purpose of navigation support for vessels operating in polar waters. One of the important principles in ice navigation is maintaining the freedom of maneuver to avoid the risk of vessels being trapped and trailing behind the ice [6]. According to [6], it is stated that non-ice-strengthened ships moving at a speed of 12 knots can be compromised when engaging heavy concentrations of light sea ice conditions. Therefore, accurate sea ice detection is crucial when assessing and adhering to operational limits. There is an emerging need for onboard ice navigation decision support systems as the number of commercial vessels in polar seas grows faster than the availability of icebreakers and ice specialists.

Sea ice information is mostly made available in ice charts by combining the data acquired from satellites, aerial, shipboard observations, and in-situ sensors. Earth observation sensors such as Synthetic Aperture Radar (SAR), imaging radiometers (visible-infrared sensor and microwave sensor), scatterometers, and altimeters are used [7] for remote detection of sea ice. Many researchers [8]–[10] use data obtained via such sensors to develop models and algorithms for ice detection and analysis. Although these data provide critical information for the safe planning of marine operations, on-site tactical navigation decisions in ice-covered waters highly rely on visual information to assess the operational risk posed by ice conditions in the field of view. Computer vision and image processing are fields of study where mathematical techniques are developed for computers to understand images and videos [11]. Over the years, these techniques have evolved such that objects in an image can be detected and classified (identifying the class of objects, e.g., vehicles, humans, etc.).

1.1.2 Problem Definition

Traditional computer vision techniques for real-time object detection are inefficient when small environmental changes are inevitable. This is a significant concern considering the challenging environment in polar seas, as there can be varying lighting conditions, glare on ice and water caused by the sun, and different ice forms. Compared to traditional algorithms, the deep learning approach in computer vision facilitates end-to-end learning that dismisses

the manual feature definition, extraction, and matching. With the increased popularity of deep learning, many neural network-based semantic segmentation algorithms have been introduced over the years, outperforming traditional algorithms in accuracy, robustness, and efficiency [12]. The closest work on sea-ice detection can be found in [13]. The main issues with the implementation for sea-ice detection in [13] are the high inference time and the inability to run on edge devices used in the field (e.g., embedded deep learning hardware, smartphones). Therefore, in addition to selecting accurate deep learning models for sea-ice detection, the ability to export into an edge device-compatible version of the model and a proper hardware selection is essential for real-time computation.

1.1.3 Objectives & Contributions

The study is sorted into the following tasks to achieve the sea-ice detection objective using deep learning methods.

Task 1: Select and train a state-of-the-art NN semantic segmentation model for sea ice detection, which can be deployed to meet the accuracy-speed trade-offs of edge computing devices.

Task 2: Comparative performance evaluation of the proposed NN model with state of art sea ice detection neural networks and datasets [13].

Task 3: Accuracy-speed evaluation of the proposed NN model on Linux-Jetson and Android-smartphone edge computing devices.

1.2 LZ detection using point cloud semantic segmentation

1.2.1 Motivation

Unmanned aerial vehicles (UAVs) have become popular in several application domains, especially in the case of multi-rotor *Vertical Take-Off and Landing Vehicle* (VTOL) vehicles with their capability to operate in constrained and unstructured environments. VTOL autonomous capabilities are particularly useful in hazardous and hard-to-reach environments [14], where the technology can enhance 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 [14]. Equipping these vehicles with the capability to identify safe *Landing Zone*s (LZs) will enable VTOLs to operate autonomously in unknown environments, ensuring the autonomous navigation system's overall safety and reducing the pilot's workload.

Autonomous identification of safe landing zones by a VTOL is a challenging task. Different approaches were introduced over the years to tackle this challenge using various sensor data. A safe landing zone is considered a low gradient, obstacle-free, open support surface with sufficient area for the VTOL vehicles [14]. Due to the rule-based nature of classical mathematical modeling, they lack the ability to learn and fine-tune new data that becomes available for LZ detection purposes. Neural network-based landing zone detection is a recent development that offers several advantages over classical methods. *Neural Network*s (NNs) can learn complex patterns related to LZ identification using labeled training data, provided 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 GPU acceleration using new machine learning libraries such as TensorFlow and PyTorch.

1.2.2 Problem Definition

Computer vision-based *Convolutional Neural Network* (CNN) algorithms were developed for image segmentation, object detection, and tracking to detect the possible safe landing zones. Although CNN-based object detection and classification are well-established, vision data do not have the reliable depth information required to detect LZ accurately. Also, the inability to register data in poor lighting conditions, irregular sensor sensitivity, and limited *Field of view* (FOV) make the metric reconstruction of the environment difficult. Globally registered *Light Detection And Ranging* (Lidar) point clouds can be used for landing sites to determine the probability of the safety prediction for each LZs. With the wide availability of aerial Lidar data and development in artificial neural networks, recent research studies have focused on using CNNs-based point cloud semantic segmentation to classify terrains and identify suitable LZs [15]. Additionally, NN methods are advantageous due to their adaptability to different data sources, ability to train as a general model applicable for many datasets, and not requiring parameter tuning to different scenarios in data. This study evaluates state-of-the-art

NN models for LZs detection using Lidar data and the feasibility of efficiently incorporating the NN model into a *Visual Lidar Odometry and Mapping* (VLOAM) pipeline. VLOAM is a real-time point cloud map generation method that can be used as a supporting module when implementing landing zone evaluation in real-time.

1.2.3 Objectives and Contributions

The study considers the following tasks as the objectives of the problem of LZs detection using point cloud semantic segmentation and the expected contributions.

Task 1: Comparative evaluation of point-based and projection-based NN models for the capability of handling different Lidar data sources and generating accurate predictions compared to classical rule-based LZ selection algorithms.

Task 2: Develop an efficient architecture and hyperparameters to integrate NN LZ evaluation in VLOAM map generation modules.

Task 3: Performance testing of the proposed LZ detection method for accuracy and speed on onboard hardware suitable for field deployment.

1.3 Organization of the Thesis

This section outlines how the thesis is organized.

Chapter 1: Introduction presents an overview of the two studies on deep learning for field robotic applications. The overviews elaborate on each study's motivation, problem definition, and objectives.

Chapter 2: Background presents a literature review of similar or related work on sea-ice detection and LZs detection.

Chapter 3: Sea-ice detection using DeepLab discusses the performance of the state-of-the-art DeepLabv3 for sea-ice detection compared with a base model and deployment feasibility on navigation modules and edge-device applications.

Chapter 4: LZ detection using projection-based methods presents the evaluation of projection-based point cloud semantic segmentation models for their performance in the accuracy-runtime trade-off.

Chapter 5: LZ detection using point-based methods introduces how point-based point cloud semantic segmentation models can be utilized for LZ detection for VTOLs. The chapter will show how a point-based method can be tuned to achieve an accuracy-runtime trade-off for online LZ detection in a *Robot Operating System* (ROS)-based VLOAM pipeline.

Chapter 6: Conclusion presents the conclusions from the conducted studies, identified future improvements, and the contributions made by the author.

1.4 Statement of Co-authorship

I hereby declare that this thesis incorporates material from joint research, as mentioned in the following paragraphs.

Chapter 3 of the thesis includes the outcome of the publication at the OCEANS 2021 conference, co-authored with Benjamin Dowden and Jesse Chen and supervised by Dr. Oscar De Silva and Dr. Weimin Huang. In all cases, only my primary contributions towards this publication are included in this thesis, and the contributions of co-authors: Benjamin Dowden with his work in [13] and Jesse Chen through his contribution to the development of a mobile application with deep learning inference for sea-ice detection.

Chapter 4 incorporates research findings published at NECEC 2021 conference supervised by Dr. Oscar De Silva and Dr. Awantha Jayasiri. In all cases, the key ideas, primary contributions, experimental designs, data analysis, interpretation, and writing were performed by myself.

Chapter 5 of the thesis includes published material co-authored with Sachithra Atapattu and supervised by Dr. Oscar De Silva, Dr. Awantha Jayasiri, Dr. George Mann, and Dr. Raymond Gosine. In all cases, only my primary contributions towards this publication are included in this thesis, and the contributions of co-author, Sachithra Atapattu was related to the classical geometric method development for point cloud segmentation for LZ detection. This chapter also incorporates unpublished research work where the primary contribution and experimental results were performed by myself. I certify that I have properly acknowledged the contribution of other researchers to my thesis and have obtained permission from each co-author(s) to include the above material(s) in my thesis.

This thesis includes 4 original papers that have been previously published/submitted to conferences and journals for publication, as shown in Table 1.1.

Table 1.1: Contributions towards original papers

Thesis Chapter	Publication Title/ Full Citation	Publication Status
Chapter 3	Balasooriya, N., Dowden, B., Chen, J., De Silva, O. and Huang, W., 2021, September. "In-situ Sea Ice Detection using DeepLabv3 Semantic Segmentation." In OCEANS 2021: San Diego–Porto (pp. 1-7). IEEE.	Published
Chapter 4	Balasooriya, N., De Silva, O., Jayasiri, A., 2021. "Comparison of point cloud semantic segmentation models for SLAM" NECEC 2021	Published
Chapter 5	Atapattu, S., Balasooriya, N.M., De Silva, O., Jayasiri, A., Mann, G. and Gosine, R., 2020. "Landing Zone Identification Using a Hardware-accelerated Deep Learning Module." 77th Annual Forum & Technology Display. The Vertical Flight Society.	Published
	Balasooriya, N.M., De Silva, O., Jayasiri, A., Mann, G., 2022. "AI-based Landing Zone Detection for Vertical Takeoff and Land LiDAR Localization and Mapping Pipelines." Drone Systems and Applications, Canadian Science Publishing.	Under reviews

Chapter 2

Background

This chapter will introduce a literature review on sea-ice detection using image semantic segmentation, LZ detection using point cloud semantic segmentation, and an introduction to transfer learning and the performance metrics used to evaluate the models discussed.

2.1 Image Semantic Segmentation

Computer vision and image processing are fields of study where mathematical techniques are developed for computers to understand images and videos [11]. Over the years, these techniques have evolved such that objects in an image can be detected and classified (identifying the class of objects, e.g., vehicles, humans, etc.).

Image segmentation is where an image is sectioned into regions and structures like lines, curves, circles, shapes, edges, etc. But these outputs do not give meaningful information regarding what is rendered in images. In semantic segmentation, the objects and scenarios in images are understood at a pixel level, where each pixel is given a label belonging to an object class. Image semantic segmentation can also be defined as an extension of image classification where the model pinpoints the location of a corresponding object by outlining its boundary. The outlining of the edge of an object in an image is called masking. Most semantic segmentation algorithms are an encoder-decoder network where the encoder encodes a latent space representation of the image, and the decoder decodes it to form segment maps outlining object location.

However, most traditional methods cannot distinguish different objects in an image. The following sub-sections will briefly overview both conventional algorithms and the CNN-based

methods and their advantages and limitations.

2.1.1 Traditional Algorithms

Before the introduction of CNNs, traditional image processing algorithms were used to understand the images at a pixel level. These traditional methods can be defined into several categories based on the techniques used to understand the images, as illustrated in Fig 2.1.

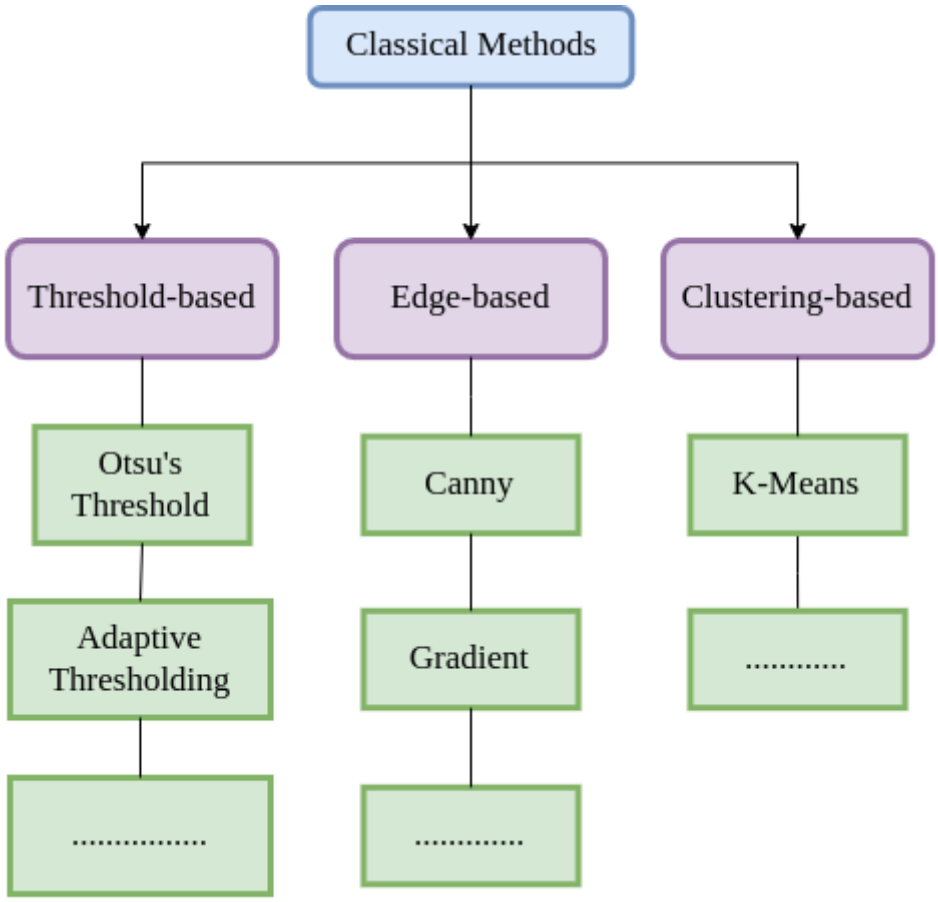


Figure 2.1: Sample categorization of traditional algorithms

For thresholding, the input image should be grayscale or converted to grayscale for processing. In simple thresholding, the images are processed into segments, where each pixel is divided into two classes based on a defined threshold value. In this scenario, a global single threshold value is used, which is not advantageous in changing light conditions. Adaptive thresholding determines the threshold for a pixel based on a small region around it, giving different thresholds for different regions of the same image. Otsu's Threshold method [16] processes the image histogram of a grayscale image, segmenting the objects by minimization of the variance on each of the classes and determining the optimal threshold based on this

histogram.

Edge-based segmentation is also called edge detection, where it classifies which pixels in the image are edge pixels and labels them in a separate class. The gradient is the change in the gray level of an image with direction, which can be calculated by taking the difference in the value of neighboring pixels. Vertical edges can be detected using a horizontal gradient operator, while a vertical gradient detector can identify horizontal edges [17]. Canny edge detection [18] is a multi-stage algorithm. The first step is the noise reduction in images using a Gaussian filter, followed by the calculation of the intensity gradient of the image as the second step. In the third step, non-maximum suppression removes unwanted pixels. Finally, using two threshold values, hysteresis thresholding is applied to detect the real edges.

Compared to threshold-based and edge-based image segmentation methods, clustering performs better in generating promising segments in an image. The k-means clustering algorithm is a widely used unsupervised algorithm that uses all the pixels and clusters them based on common attributes.

Though traditional computer vision techniques like gradient vector flow snake algorithm and support vector machines classifiers are used in [19]–[21] to classify and detect sea-ice and ice floes, the works are based on satellite images. In [22]–[24], traditional computer vision algorithms such as thresholding, k-means clustering, and spatial filtering are used on imagery captured by shipborne cameras to detect ice types and concentration. Traditional computer vision techniques for real-time object detection have proven inefficient when inevitable environmental changes occur. This is a significant concern considering the extreme environment in polar seas, as there can be varying lighting conditions, glare on ice and water caused by the sun, reflections on water, and the different forms of ice.

2.1.2 State-of-the-art CNN methods

Compared to traditional algorithms, deep learning methods in computer vision facilitate end-to-end learning, which circumvents the process of manual feature definition, extraction, and matching. With the increased popularity of deep learning, many neural network-based semantic segmentation algorithms were introduced over the years, which have outperformed traditional algorithms in accuracy, robustness, and efficiency [12]. This section will briefly overview the state-of-the-art methods for image semantic segmentation and work related to

sea-ice detection using deep learning-based networks.

Several image semantic segmentation challenges were introduced over the years to evaluate different NN-based models developed to detect target object classes. PASCAL VOC 2012 challenge [25] is one of the earliest challenges developed for object detection and segmentation. The dataset contains more than 11,000 images in train and validation sets, while 10,000 images are set for testing. The *Common Objects in COntext* (COCO) dataset challenge [26] for object segmentation contains more than 200,000 images with over 500,000 object instances in 80 categories. The Cityscapes dataset [27] was released in 2016, containing complex segmentations of urban scenes from 50 cities. The images are fully segmented into 29 object classes within eight super-categories: flat, human, vehicle, construction, object, nature, sky, and void.

Fully Convolutional Network (FCN) [28] is the one of the first fully convolution-based NN model end-to-end trained for image semantic segmentation. The FCN takes an image with an arbitrary size as input and outputs a segmented image with the same size while replacing fully connected layers with convolutional layers. The authors of [28] recorded a 62.2% mIoU score on the PASCAL VOC 2012 segmentation challenge. U-Net [29] is an extension of the FCN architecture developed for microscopic biological images. The U-Net model has two parts: the contracting part computes features, and the expanding part uses deconvolution to reduce the number of feature maps and spatially localize patterns in the image. The most notable characteristic of the U-Net model is it does not use any fully-connected layer. U-Net architecture is extended in the PSPNet and Deeplabv3 models, which will be described in later chapters. Mask-*Region-based Convolutional Neural Network* (R-CNN) is introduced as a state-of-the-art model for semantic segmentation on the COCO challenges.

Though a large body of work is done using deep learning algorithms for sea-ice detection, most of them are based on satellite imagery [30]–[32]. Using satellite imagery has several limitations due to remoteness, year-round operational restrictions due to weather conditions in polar regions, low reliability of satellite communication at higher latitudes, and operation costs. As a result, this study focuses on using visual data images captured by a shipborne camera for real-time ice detection.

Closely related to the work in this study includes [33] where authors proposed a CNN-based semantic segmentation model for river-ice detection from the drone footage of the yellow river in China. Other related works include [34], [35], where authors have utilized images

captured on board a cruise for sea-ice object classification. Their work focused on classifying sea ice images into nine ice categories. In [36], the authors have utilized a modified U-Net [29] model for the segmentation of ice objects where the encoder is pre-trained on the ImageNet dataset [37]. The methods [34]–[36] manually exclude the vessels' bows from the image data by defining the region of interest for the segmentation task. The report does not describe the comparative performance of their selected model on any standard evaluation metrics. Additionally, the manual selection of regions of interest to extract the ice-containing segments from the images in [36] limits the application of their solution to real-time sea-ice detection and implementation on different hardware configurations. [38] can be considered a promising development of deep learning-based sea-ice detection using shipborne camera imagery. In their work, the authors disclosed the development of a custom 3D camera system on board a vessel for continuous monitoring of ice and water and experiments carried out to verify their camera system. The developed system focuses on a patch of ice on the vessel's port side for continuous 3D topography reconstruction and uses an image processing scheme for sea ice detection. However, this method is only applicable for downward view angles from the side, which is not considered in this thesis.

2.2 Point Cloud Semantic Segmentation

A point cloud is a set of data points representing objects or space in the X, Y, and Z geometric coordinate system. Point clouds can be generated by photogrammetry software using camera images, 3D laser scanners, *Synthetic Aperture Radar* (SAR) systems, and Lidar. This study uses point clouds generated by Lidar sensors; thus will only review the related research. Based on the Lidar sensor type and the platform, the point density can vary from 10 points per square meter (pts/m^2) (sparse point cloud) to thousands of pts/m^2 (dense point cloud) [39].

Point cloud segmentation and point cloud semantic segmentation have different meanings. Point cloud segmentation methods group points with similar geometric or spectral characteristics, while point cloud semantic segmentation associates each point in the point cloud with a semantic label. This section will brief both point cloud segmentation and point cloud semantic segmentation methods introduced over the years and work related to landing zone detection using each approach.

2.2.1 Classical Methods using Geometric Features

In this study, point cloud segmentation using geometric features will be called classical methods. These classical methods process point clouds by grouping raw 3D points into non-overlapping regions corresponding to a specific structure or object. This grouping of points is done mainly using handcrafted features generated from geometric constraints and statistical rules.

Edge-based classical methods use an approach similar to edge detection in 2D images. As the shapes of the objects can be distinguished using their edges, point clouds can be segmented based on the points close to edge regions. A rapid change in intensity in the data can be used to locate such points [40]. Work in [41]–[43] uses different edge-based algorithms for point cloud segmentation. The region-growing method combines features from two points or two regions to measure similarities among 3D points or 3D voxels and merges if they meet the spatial distance constraint or similar surface properties. Three main factors should be considered when developing a region-growing algorithm: similarity measurement, growth unit, and seed point selection [44]. The similarity measurement can be done using euclidean distance or normal vectors. Popularly used methods for growth unit factors are the k-d tree, voxel-based region growing [45], and adaptive octree [46].

The base concept of model fitting is to match the point cloud to different primitive geometric shapes. Most model-fitting algorithms are developed on Hough Transform, and *Random Sample And Consensus* (RANSAC) [47]. RANSAC is widely used in plane segmentation methods as it does not require complex optimization or ample memory resources and efficiency in detecting more objects [48].

Early work on using simulated Lidar for landing zone detection is proposed in [49], which uses a relatively simple geometric analysis of the terrain roughness and slope. In [50], the authors use digital elevation maps from NASA data explorer and apply a modified quadtree algorithm to find potential landing sites. In [51], [52], landing site detection algorithms using *Principal Component Analysis* (PCA) are proposed based on the terrain information retrieved using Lidar data. Authors of [53] devised an algorithm for safe landing zone detection using PCA and a variation of RANSAC called *Progressive Sample Consensus* (PROSAC) algorithm. One of the state-of-the-art approaches using LiDAR point clouds is a coarse evaluation of LZs based on slope and roughness, as presented in [14], which is also similar to work in [49]. Authors of [14] have conducted experiments for safely landing a helicopter on identified zones and mentioned that this method can take data at a rate of 17 ms per 100,000 points and

compute coarse evaluation at a rate of 699 ms per 100 000 cells on an Intel core 2 processor.

Even though these classical 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 fine-tune new data that becomes available for LZ detection purposes. Neural network-based landing zone detection is a recent development that offers several advantages over classical methods. NNs can learn complex patterns related to LZ identification using labeled training data, provided 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 a graphics processing unit (GPU) acceleration using new machine learning libraries such as TensorFlow and PyTorch. In [54]–[56], computer vision-based convolutional neural network(CNN) algorithms for image segmentation, object detection, and tracking were developed to detect the possible safe landing zones. Although CNN-based object detection and classification using images is well-established, vision data do not have the reliable depth information required to accurately detect LZs. Also, the inability to register data in poor lighting conditions, irregular sensor sensitivity, and limited FOV make the metric reconstruction of the environment difficult. Two mainstream NN-based approaches are used for point cloud semantic segmentation: projection-based and point-based architectures. The next two sections will brief both approaches and their use in landing zone detection.

2.2.2 Projection-based Semantic Segmentation

Projection-based semantic segmentation methods transform the 3D point cloud by projecting it onto a 2D image using different intermediate representations. Work in [57] projected a 3D point cloud onto 2D planes using multiple virtual camera views and applied multi-stream FCN to predict pixel-wise scores. In [58], the authors generated *Red Green Blue* (RGB) and depth snapshots of a point cloud using multiple camera positions and applied 3D segmentation for predicting pixel-wise labels. Assuming that point clouds are sampled from Euclidean surfaces, authors in [59] projected the local surface geometry around each point to a virtual tangent plane. Tangent convolution is then applied to this surface geometry. But, these multi-view semantic segmentation methods are sensitive to viewpoint selection and occlusions and were not fully explored due to information loss during projection.

Work in [60], one of the leading articles in point cloud classification, has proposed a CNN-

based pipeline, "SqueezeSeg." Their approach transfers the LiDAR point cloud data into dense, 2D grid representation using spherical projection and fed into a 2D CNN. Further improvements to the SqueezeSeg network were proposed in [61], [62] to enable a runtime faster than the sensor rate, i.e., 10Hz. Though this was an added benefit for real-time processing, the SqueezeSeg framework only uses the frontal 90 degrees of the scan, which does not capture the full Lidar scan. Benefiting from the introduction of spherical projection in [60], RangeNet++ [63] was introduced, which uses a DarkNet backbone to process a range-image generated by the projection of a point cloud. The semantic labels generated by the RangeNet++ model are first transferred to a 3D point cloud, and GPU-enabled k-nearest neighbor search is applied to retrieve labels for all points in the cloud. This diffuses the problem of discretization errors and blurry inference outputs. Work in [64] uses the similar spherical projection of point cloud data into a 2D range-view image and applies an encoder-decoder-like network for point-wise classification score generation.

These methods delivered a lower accuracy than point-based methods but were significantly faster in processing. The candidate models for the projection-based point cloud semantic segmentation are selected based on their performance on SemanticKITTI [65] benchmark leaderboard for single scan evaluation and the reproducibility of the code and the results. Although speed improvement is possible with the projection-based network, the accuracy of these methods is still questionable, especially for safety-critical decisions like detecting landing zones. 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 techniques also require a method of point cloud aggregation before range image generation for reasonable results. In safety-critical applications, high accuracy and precision in identifying safe LZs are more pertinent than the detection speed.

2.2.3 Point-based Semantic Segmentation

Point-based semantic segmentation methods directly work on irregular, unstructured point clouds. Voxelization represents each point in the point cloud as 3D voxels, which is another method used in point cloud processing [60], [66]. In [66], a 3D CNN architecture called "Voxnet" based on volumetric occupancy grid representation of point clouds is described. The 3D convolution layers in their work are applied to these voxel grids. They have implemented the Voxnet for various configurations, and for the slowest configuration, it has taken 6ms for classification using a Tesla K40 GPU. However, voxelization of point clouds

introduces additional computational overhead, and fine details of the original data can be lost in the process [67].

Work in [68] proposes an approach termed "PointNet," which feeds unordered point clouds directly as the input to the algorithm. The per-point features are extracted using shared *Multi-Layer Perceptron*s (MLPs), and global features are learned using symmetrical pooling functions. PointNet architecture has achieved improved performance for semantic segmentation of point clouds over the traditional methods proposed in [66] and [69]. A point-wise pyramid pooling module for semantic segmentation of point clouds is discussed in [67], which uses two-direction hierarchical *Recurrent Neural Network*s (RNNs) to incorporate long-range spatial context. This approach outperforms the baseline PointNet method in [68].

Improvements to PointNet were suggested in PointNet++ [70], where the model clusters the points hierarchically and progressively learns from large local regions. The multi-scale and multi-resolution grouping was introduced to overcome the varying density of point clouds. PointNet and PointNet++ successfully delivered higher overall accuracy but were too slow in processing large-scale point clouds. Additionally, the performance of the above two models in detecting vegetation, buildings, and vehicles was not up to the expected level for our problem. RandLA-Net [71] used a downsampling method to remove features at random and then a feature aggregation module to increase the receptive field for each 3D point where geometric details are preserved. *Kernel Point Convolution* (KPConv) [72] introduced a novel kernel point convolution method to process point clouds directly without intermediate computations. Though these two methods delivered a promising performance, their architectures were not developed for a continuous input of point cloud data. Authors of [73] have proposed an approach termed "ConvPoint" to process the unstructured point cloud data using a continuous convolution formulation. Through this continuous convolution formulation, they have designed neural networks similar to 2D CNNs to process the point cloud data as 3D data. Their results proved the flexibility of this approach for different architectures. Moreover, at the time of writing this thesis, the ConvPoint model has the highest accuracy and fast execution for semantic3d point cloud segmentation benchmark [74], which involves eight classes and supports both Lidar and color data of the point cloud. According to the ConvPoint paper, [73], the suggested model has achieved an overall accuracy of 93.8% and a mIoU of 75.0% on the Semantic3D dataset.

2.2.4 Discussion on point cloud semantic segmentation methods

Due to the direct processing of point cloud data in point-based semantic segmentation methods, they are at a disadvantage of memory overflow and lower runtime speed in real-time applications. But, compared to projection-based methods, point-based methods can deliver a highly accurate and precise performance, which is a critical requirement for autonomous landing zone detection problems. This requirement overpowers projection-based networks and demands a runtime optimization for point-based networks. Both methods were found to not carry sufficient color information for accurate use of texture in the semantic decision, as a colored point cloud is an order of a magnitude lower in resolution when compared with a camera image of the same viewpoint. From our investigations thus far, a neural network applied to the image would be more suitable for using color information and fusing with the point cloud segmentation.

Therefore, a novel landing zone detection architecture will be proposed in this thesis to address the mentioned issues. In this architecture, a point-based point cloud semantic segmentation model with a better runtime-accuracy trade-off will be used for generating LZ labels, while image segmentation and object detection will be employed as an additional component to project the resulting pixel labels on a point cloud. This thesis focuses on integrating a deep neural network-based LZ detection module into 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. Applying deep NN for LZ detection has the advantages of incremental and adaptive learning ability from new data, faster runtime, and a graphical processing unit for computations that can free the central processing unit (CPU) for other tasks. It can also ease fusing different NN networks and code optimization capability compared to classical methods. The top-performing point-based semantic segmentation model in the Semantic3D leaderboard will be evaluated for its performance on three variations of point cloud data to achieve this objective.

2.3 Performance Metrics

Mean Intersection over Union (mIoU), accuracy, average runtime, and throughput are used as the metrics to evaluate the performance of different NN models. This section will briefly overview each metric.

2.3.1 Intersection over Union

Intersection over Union (IoU), 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 2.2 shows a generic visualization of IoU while Eq. (2.1) is the mathematical computation of IoU. The mIoU is the calculation of the IoU of each class and averaging the values.

$$\text{IoU} = \frac{|A \cap B|}{|A \cup B|} \tag{2.1}$$

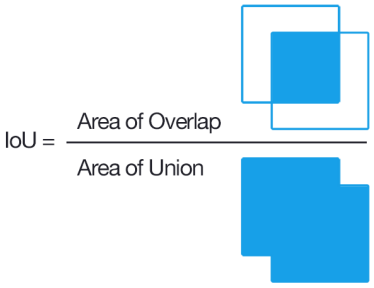


Figure 2.2: IoU calculation visualization. Source: Wikipedia

2.3.2 Accuracy

Informally, accuracy is the fraction of correct predictions over the total number of predictions assessed by a model. Equation 2.2 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 that most data has a class imbalance, where some object classes can dominate the point cloud while others make only a smaller portion. This issue cannot be ignored in a real-world navigation setting.

$$\text{Accuracy} = \frac{\text{Number of correct predictions}}{\text{Total number of predictions}} \tag{2.2}$$

2.3.3 Average runtime

The average runtime speed is the average measure of time taken for the model to compute inference over a set of point cloud scans. The average runtime speed is calculated as shown in Eq. (2.3)

$$\text{Average runtime} = \frac{\sum_{n=1}^{k} t_n}{k} \tag{2.3}$$

where, k = total number of scans and t_n = inference time of n^{th} scan.

2.3.4 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. The number of points processed during a unit time of seconds is calculated, known as throughput. This is computed using the Eq. (2.4),

$$\text{Throughput of } n^{th} \text{ scan} = \frac{\text{Total number of points in } n^{th} \text{ scan}}{\text{inference time of } n^{th} \text{ scan}}, \tag{2.4}$$

and averaged over the total throughput values.

Mean-IoU, accuracy, and inference time metrics will be used to evaluate the performance of the NN model for sea-ice detection. While mIoU, accuracy, average runtime, and average throughput are used to assess the point cloud semantic segmentation models for LZs detection.

Chapter 3

Sea-ice detection using Deeplab

This chapter will describe the extended study on previous work [13] by evaluating the performance of the Deeplabv3 [75] model for sea-ice semantic segmentation with the benchmark model *Pyramid Scene Parsing Network* (PSPNet101) from [13], and assess the compatibility of deploying the system on embedded and mobile hardware platforms for real-time computation. This study focuses on evaluating the inference speed of the Deeplabv3 model compared with PSPNet101 using both Nvidia Jetson AGX Xavier and Android mobile computing platforms and validates Deeplabv3 as a more compatible and enhanced architecture for the purpose.

The sections in this chapter will briefly introduce the benchmark PSPNet101 model and Deeplabv3 model, the methodology, experimental results on different hardware platforms, and a discussion of the results obtained.

3.1 Background

To achieve the objectives of this study, the proposed benchmark model in [13], PSPNet101 is selected as the state-of-the-art model for sea-ice semantic segmentation; and comparatively evaluated with Deeplabv3, which outperformed PSPNet101 [75] on PASCAL VOC 2012 benchmark [76] and performed equally on Cityscapes benchmarks [27].

3.1.1 PSPNet101

PSPNet101 was introduced for semantic segmentation as a competitor in ImageNet Scene Parsing Challenge 2016. The PSPNet101 architecture captures the global context of the whole image when predicting the objects at a pixel level. The encoder of the PSPNet101 has

dilated convolutions [77] at the last layers of the CNN backbone, which deliver rich features to the pyramid pooling module. The pyramid pooling module in part (c) of Fig. 3.1 fuses these rich features in four levels of pyramid scales to cover the different sizes of portions of the image, and each is passed through a convolutional layer. The output is then up-sampled to the same size as the feature map and concatenated with the features from the CNN backbone to be passed into the decoder to form the final feature representation. The decoder takes the concatenated features that contain local and global scene-level information and gets the per-pixel predictions by passing them through convolutional layers. A simplified graphical overview of the PSPNet101 architecture can be visualized as in Fig. 3.1.

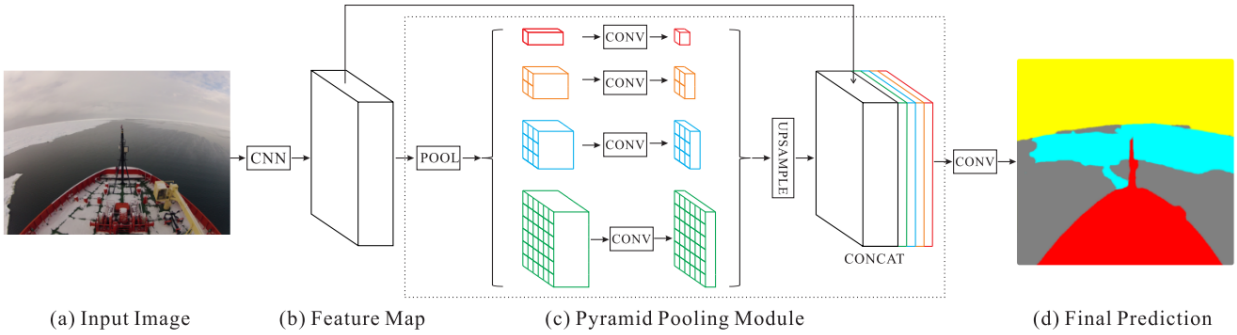


Figure 3.1: Overview of the PSPNet architecture. [78]

According to [78], it has a recorded performance of mIoU of 85.4 on PASCAL VOC 2012 benchmark [76] and an accuracy of 80.2% on Cityscapes benchmarks [27]. For the purpose of this study, the Keras implementation of PSPNet fine-tuned on the Cityscapes dataset [13] is used.

3.1.2 Deeplabv3

Deeplabv3 is an enhanced version of its predecessor, Deeplab [79], in which the architecture is augmented to capture longer-range information. Deeplabv3 network adopts several neural network architectures to extract features from the input image, which are referred to as backbones. The performance of the semantic segmentation model highly relies on the features extracted by these backbone architectures [80]. To achieve the goal of computational efficiency, Tensorflow implementation of Deeplabv3 with Mobilenetv2 [81] backbone was used as it reports a performance similar to that of other backbone architectures while having faster inference and compatibility with mobile applications. Figure 3.2 illustrates the Deeplabv3 architecture. In general Deep Convolutional Neural Networks (DCNNs), the spatial resolution of feature maps is reduced by the max-pooling layers in convolution. To

recover the spatial resolution, DCNNs use deconvolutional layers with consecutive striding, which can result in information loss. Instead, the Deeplabv3 model uses atrous convolution to control the feature map resolution and keep the stride constant. Like the PSPNet101 pyramid pooling method, Deeplabv3 uses atrous spatial pyramid pooling [79] with different rates to capture multi-scale contextual information. The resulting concatenated features are then parsed through a convolutional layer and upsampled to keep the ground-truth values intact. With the encoder-decoder architecture, the location/spatial information is retrieved.

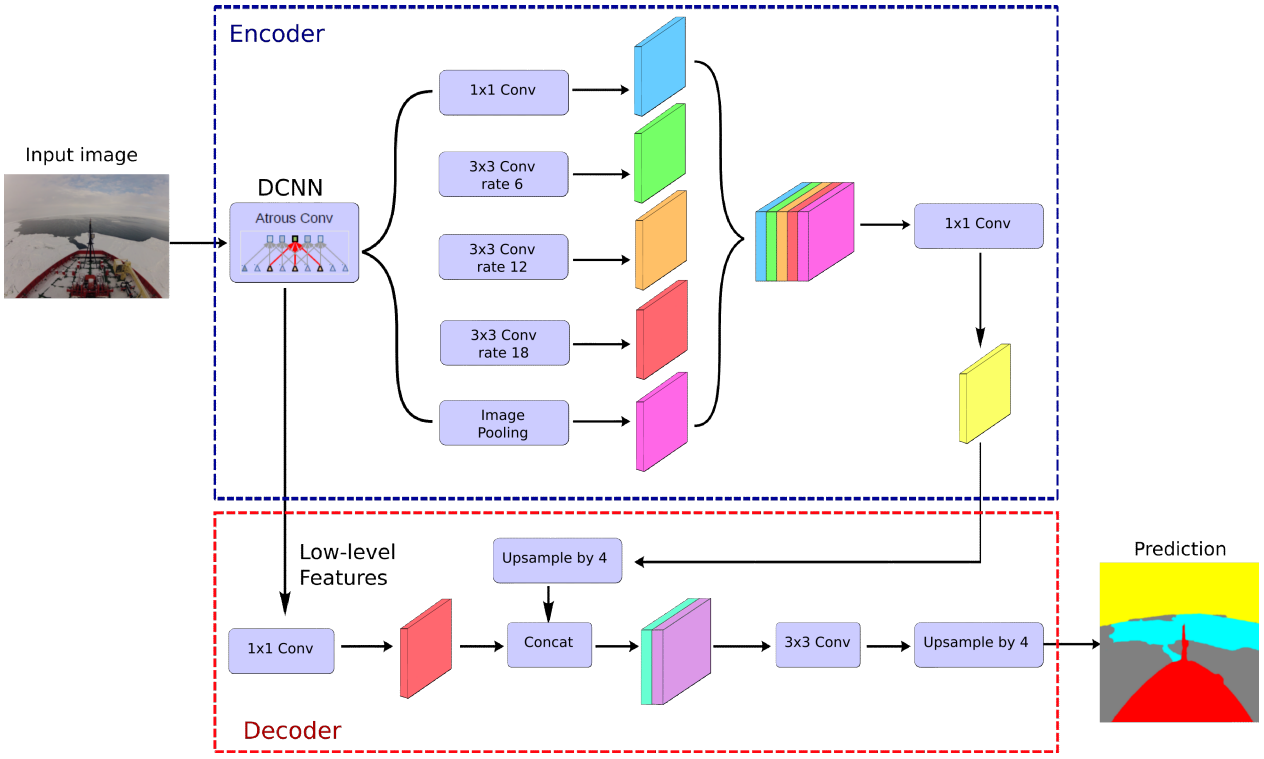


Figure 3.2: Overview of the Deeplabv3 architecture.

3.2 Methodology

This section will discuss the methodology used to train and evaluate the Deeplabv3 network for sea-ice semantic segmentation and detail the two hardware platforms used for inference testing.

3.2.1 Dataset

Data plays a major role in deep learning methods. This study's deep learning-based semantic segmentation models use supervised learning algorithms for end-to-end learning. Supervised

learning algorithms are designed to learn by example. Those examples are called training data, consisting of inputs paired with the correct output. During training, the deep learning models will learn the patterns between the input and the corresponding desired output. When the model learns those connections, it will then be tested on unseen inputs and predict the output. This training process requires a substantial amount of data for the model to learn the patterns more precisely and accurately.

The lack of abundant data on sea-ice imagery obtained on board an icebreaker was a challenge in training the deep learning models. To address this issue, in the previous work [13], a sea-ice detection dataset was created using imagery from a Go-Pro camera on board the Nathaniel B. Palmer icebreaker on its two months expedition through the Ross Sea, Antarctica. The finalized dataset contains 431 images captured on four unique days of the journey, which were labeled into four semantic classes: Ice, Ship, Ocean, and Sky. This labeling was carried out using the freely available semantic labeling tool called *PixelAnnotationTool* [82], where each pixel of the captured image is given one of the four semantic classes in numeric format. For the purpose of training, evaluation, and performance comparison, the dataset is divided into training, validation, and testing sets, as shown in Table 3.1. The training set is used to teach the model the connections between the input and the desired output, while the validation set is used to check the accuracy of the learned model during the training process. The test set is a separate set of inputs that will be used to establish the performance of the final model. Figure 3.3 shows a sample labeled image.

Table 3.1: Sea-Ice Imagery Dataset split

Split	Training	Validation	Testing	Total
No.of images	382	23	26	431

3.2.2 Transfer Learning process

For this study, the Deeplabv3 model (in section 3.1.2) pre-trained on the Cityscapes dataset was transfer-learned for the sea-ice dataset described in section 3.2.1. This approach uses a similar set of semantic labels (sky, vegetation, vehicle, etc.) with a sizable amount of training data available in the cityscapes dataset.

The transfer-learning of the Deeplabv3 model for sea-ice detection was carried out on the Google Colaboratory platform utilizing its GPU resources. Though transfer learning of

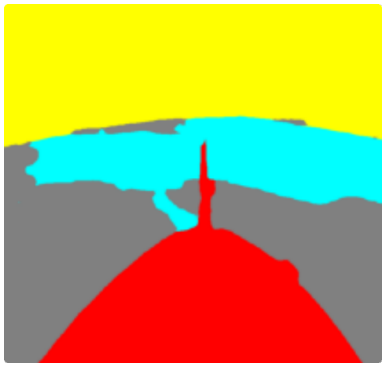
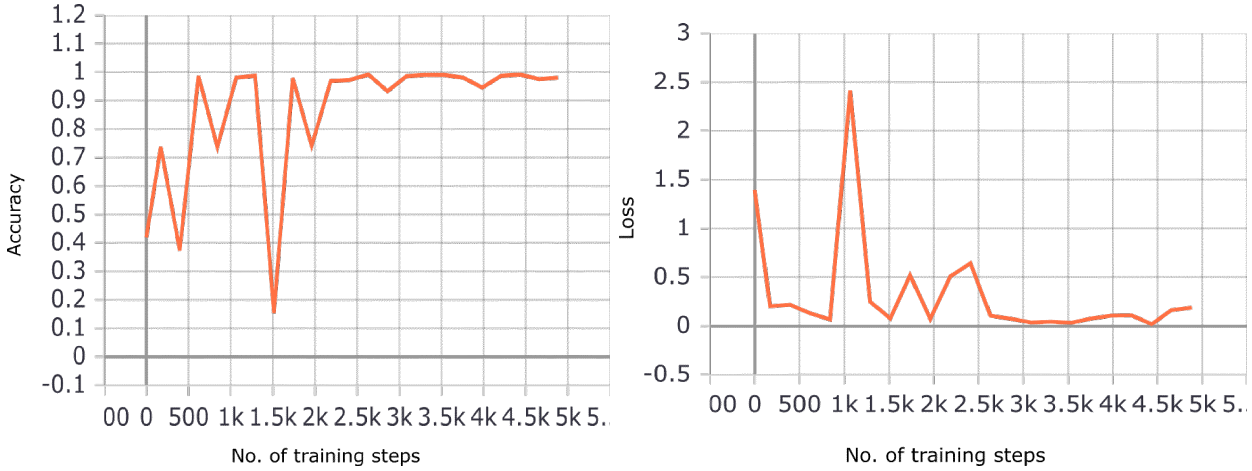


Figure 3.3: Example of a sea-ice image with the semantic labels with the four classes: ship:red, ocean:blue, ice:grey, sky:yellow ([13])

Deeplabv3 took approximately 21 minutes on Google Colaboratory, it is irrelevant for operation since the focus is inference implementation rather than transfer learning on the selected hardware platforms.

The accuracy computed on the validation set during the transfer learning process is shown in Fig. 3.4(a). The intention of teaching a neural network model is to learn from given data to minimize errors when tested on unseen data. Figure 3.4(b) interprets the prediction error over the validation set during transfer learning.



(a) Accuracy (b) Loss

Figure 3.4: Evaluation of transfer-learning performance on validation set

3.2.3 Hardware Selection

Proper hardware selection is important to deliver in-situ sea-ice detection using a ship-borne camera. Considering factors such as computational speed, availability of graphics processing

units (GPUs), and data storage capacity, an Nvidia Jetson AGX Xavier-powered navigation box was selected. Additionally, this work evaluates the capability of sea-ice detection and operational risk assessment on a hand-held device (mobile phone).

3.2.3.1 Nvidia Jetson AGX Xavier

Nvidia Jetson AGX Xavier (JAX Kit) is a state-of-the-art hardware platform for end-to-end artificial intelligence (AI) applications. The JAX kit's operational and technical specifications (as listed in Table 3.2) attest that it is the appropriate hardware for developing a navigation box. In 2021, Nvidia developers introduced an industrial version of the JAX kit, which is more compatible with an industrial environment. Here a prior generic version of JAX kit is used to validate the proposed architecture.

Table 3.2: Technical Specifications of Nvidia Jetson AGX Xavier (JAX) Developer Kit used in our work

Component	
CPU	8-core NVIDIA Carmel Armv8.2 64-bit 8MB L2 + 4MB L3 cache
GPU	NVIDIA Volta architecture 512 NVIDIA CUDA cores, 64 Tensor Cores
Memory	32GB 256-bit LPDDR4[1]
Storage	32GB eMMC[2]5.1
AI Performance	32 TOPS[3]
DL Accelerator	2 NVDLA[4]engines
Vision Accelerator	2 7-way VLIW[5]Vision Processor
No.of cameras	Up to 6 cameras
Power	10W | 15W | 30W

[1] LPDDR: Low-Power Double Data Rate
[2] eMMC: embedded MultiMediaCard - internal storage attached to the device
[3] TOPS: Tensor Operations Per Second
[4] NVDLA: NVIDIA Deep Learning Accelerator
[5] VLIW: Very Long Instruction Word

3.2.3.2 Hand-held Device

To evaluate the performance of the Deeplabv3 mobile-compatible model, it is integrated into an emulated environment of Google Nexus 6 smart mobile using Android Studio software. The hardware specifications of the emulated mobile platform are tabulated in Table 3.3. Additionally, this was tested on mobile platforms for inference performance, i.e., Samsung Galaxy S8, Samsung Galaxy Note 8, and Samsung Galaxy S21.

Table 3.3: Technical Specifications of the Google Nexus 6

CPU	2.7GHz quad-core Snapdragon 805
GPU	Adreno 420
Memory	3GB LPDDR3
Storage	32GB | 64GB
Camera	Sony Exmor CMOS 13 MegaPixels

3.3 Experimental Results

This section will assess and compare the performance of the transfer-learned Deeplabv3 model, and PSPNet101 model in [13] for the test set on the suggested navigation box (described in section 3.2.3). For the performance evaluation, two approaches were considered, i.e., evaluation using metrics and evaluation based on sea-ice detection time.

3.3.1 Evaluation using metrics

Table 3.4 compares the mIOU for each class and average accuracy over the test set in the sea-ice dataset. According to table 3.4, Deeplabv3 reported higher mIOU for the semantic classes Ice, Sky, and Ocean than PSPNet101. This suggests that Deeplabv3 performs well in distinguishing between the ice and the ocean. Additionally, the results show that PSPNet101 has outperformed Deeplabv3 in detecting the semantic class ship. This can be due to lens artifacts in several test images that Deeplabv3 was not trained to ignore. The lens artifacts have also affected the average accuracy of Deeplabv3. This issue can be resolved by including similar noisy data in the training set, which will be addressed in future work.

Table 3.4: Performance of Deeplabv3 vs PSPNet([13]) on test set

Network	IOU (%)					Avg.acc(%)
	mIOU	Ice	Sky	Ocean	Ship	
Deeplabv3[1]	**90.21**	**95.27**	**98.27**	**91.25**	81.49	95.46
PSPNet101[2]	90.1	94.7	95.7	80.8	**89.3**	**97.8**

[1] According to the results, it can be seen that Deeplabv3 has higher mIOU and IOU for semantic classes Ice, Sky and Ocean compared to PSPNet101.

[2] PSPNet101 has shown a higher IOU for detecting semantic label Ship and delivered a higher average accuracy over the whole test set.

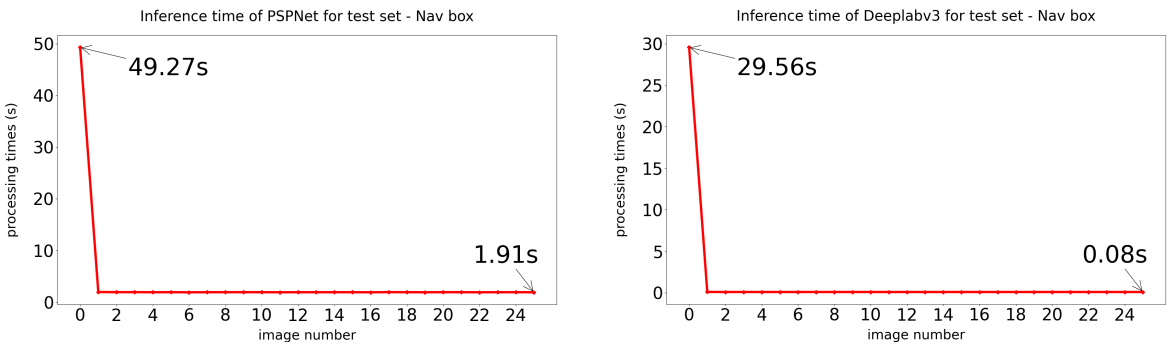
3.3.2 Inference speed

Figure 3.5 illustrates the prediction time for PSPNet101 and Deeplabv3 models for the test set on the JAX Kit. The inference on the mobile platform recorded an average prediction time of 12s. Unlike in the JAX Kit, the model is loaded during the startup of the mobile application, which was an advantage when inferring the first image.

Comparing the mIoU results listed in table 3.4, PSPNet performs similarly to Deeplabv3 regarding sea ice detection accuracy. Due to the time taken to initialize the required libraries on the nav-box platform, a relatively long processing time is expected for the first image input into the inference model (Fig. 3.5). However, when considering the inference time taken by both models on a nav-box platform, Deeplabv3 is more than 20 times faster than PSPNet101. Additionally, Deeplabv3 has mobile platform compatibility, which was successfully implemented with a mobile inference time of 12s per image (Fig. 3.7). PSPNet101 did not successfully extend to mobile device computation in the experiments. These results validate Deeplabv3 as the state-of-the-art semantic segmentation model for sea-ice detection.

3.3.3 Sample predictions

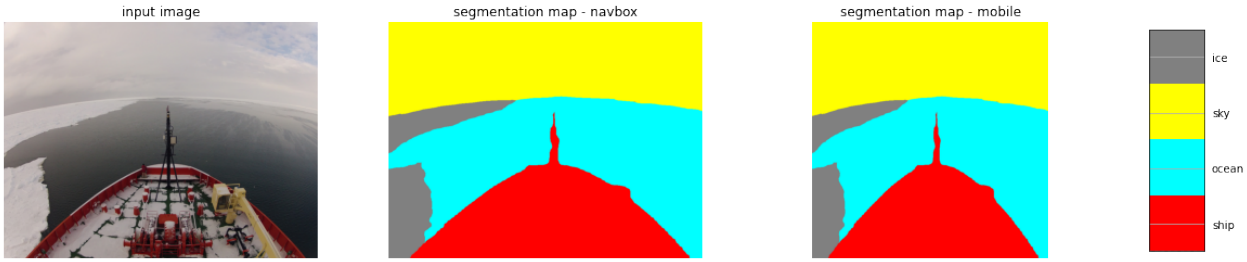
Figures 3.6 and 3.7 show example predictions of the transfer-learned Deeplabv3 model on test images when executed on both the navigation box and the mobile application. The images used to generate the sample predictions in Fig. 3.6 were taken at the same location on the



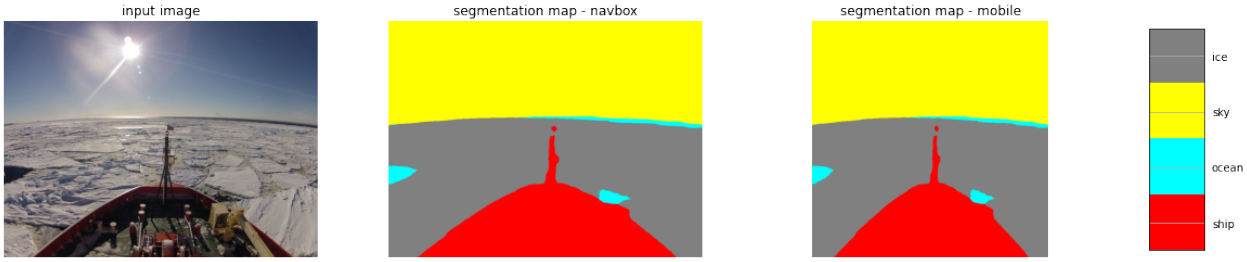
(a) (b)

Figure 3.5: Comparison of prediction time for PSPNet (3.5(a)) and DeepLabv3 (3.5(b)) on the Nvidia Jetson AGX Xavier. The excessive time for the first image includes the time required for loading the Nvidia CUDA libraries for inference purposes. But in the mobile implementation, this issue did not occur.

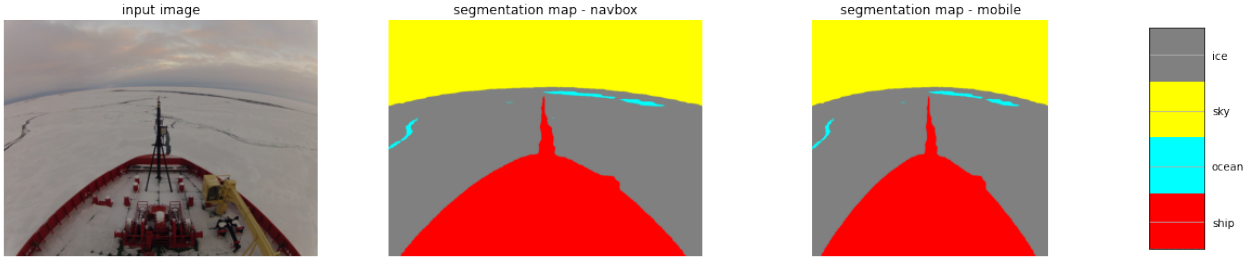
vessel as the training images. The results can drastically differ from the expected output when the model is tested on images taken from different areas of the ship. Such situations can be managed by including pictures from other locations and angles in the training set.



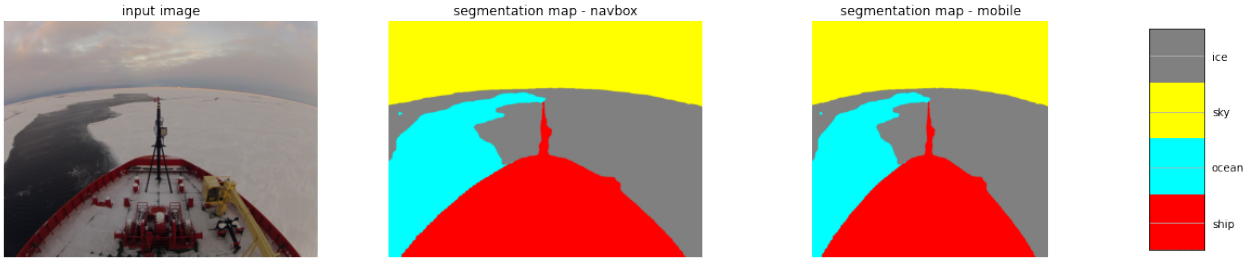
(a)



(b)

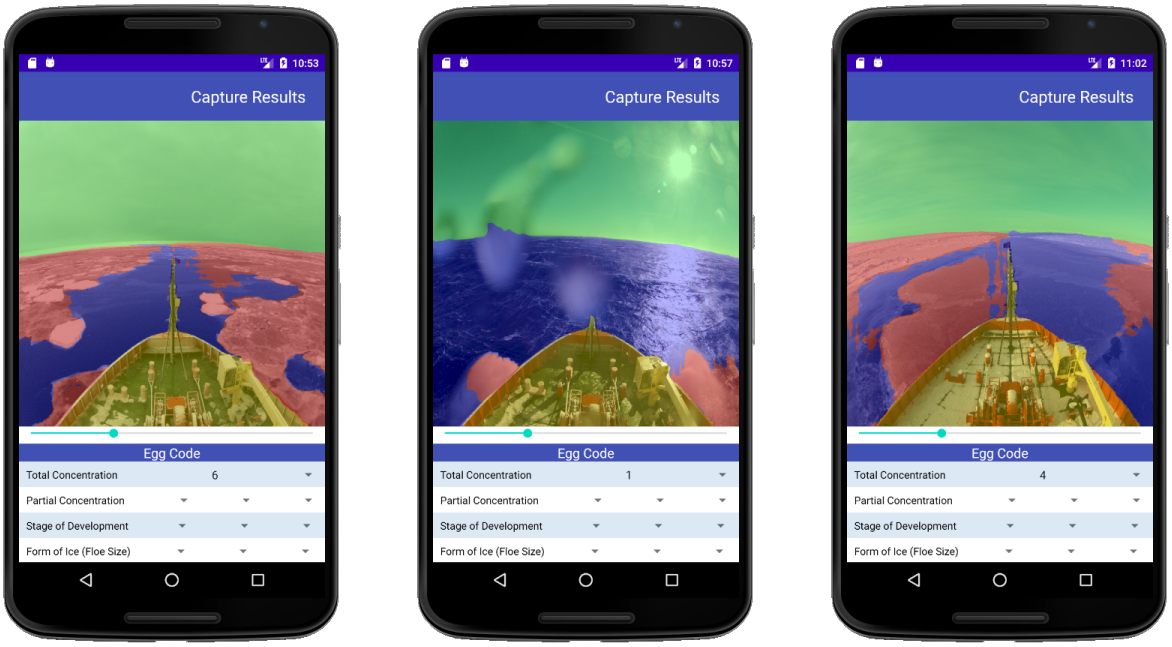


(c)



(d)

Figure 3.6: Sample predictions of the Deeplabv3 transfer-learned model. In the sample predictions, the segmentation map:nav-box is the expected output from the JAX kit, which will be used to develop the navigation box. The segmentation map-mobile is the expected prediction from the Deeplabv3 lite model, which will be implemented on a mobile device.



(a) (b) (c)

Figure 3.7: Example predictions of the Deeplabv3 model on mobile app

3.4 Discussion

This study proposes Deeplabv3 for in-situ sea-ice semantic segmentation and compares its performance with the state-of-the-art model PSPNet101. The Deeplabv3 model is transfer-learned on a sea-ice dataset and is tested and compared for its performance using the metrics per class mIOU and average accuracy. According to the conferred results, Deeplabv3 outperformed PSPNet101 in sea-ice detection with an overall mIOU of 90.21. Additionally, this report proposes the development of an onboard nav box using Nvidia Jetson AGX Xavier and a mobile application for in-situ sea-ice detection. The study compares the inference speed of both Deeplabv3 and PSPNet101 on the nav box and explores the feasibility of implementing both models on a mobile platform. Deeplabv3 required less computation time for inference on test images on the nav-box platform compared to the benchmark model, PSPNet101. Moreover, the selected backbone architecture of Deeplabv3 and its ability to be converted into a Tensorflow Lite model have accommodated the implementation of sea-ice detection on a mobile platform. However, the attempt to integrate PSPNet101 into a mobile application failed due to its lack of support for mobile platforms.

Deeplabv3 is proposed as the new state-of-the-art model for in-situ sea-ice detection, given its performance, real-time operation capability using a navigation box, and compatibility across

a mobile platform. Future developments of this work include improving the performance of the Deeplabv3 model, functionality of the model for operational risk assessment, and detection of different ice categories.

Chapter 4

Evaluation of projection-based point cloud segmentation methods for VLP-16 Lidars

As introduced in chapter 1, the ISLab of the Memorial University of Newfoundland collaborates with the NRC in developing an *Artificial Intelligence for Logistics* (AI4L) system for VTOL vehicles. This project, AI4L-112, aims to develop an AI-powered *Visual Inertial Lidar Odometry and Mapping* (VILOAM) navigation system, which can be integrated into VTOL vehicles to improve its real-time autonomy support capabilities and reduce pilot workload. VILOAM is a methodology that allows platforms to localize in an environment while simultaneously creating a reference point cloud of the surrounding using a camera, Light Detection and Ranging (LiDAR), and Inertial Measurement Unit (IMU) data. This method improves the autonomy of platforms and enables operation in unknown environments without the reliance on external positioning aids like GPS while producing informative maps. The project develops a VILOAM-capable hardware flight module with a Velodyne VLP-16 Lidar sensor.

One of the modules of this project is to develop a landing zone detection system for VTOL vehicles. A safe landing zone is considered as a low gradient, obstacle-free open surface with sufficient space to land a VTOL vehicle without compromising the safety of the personnel involved and the surrounding infrastructure [14]. These characteristics of a LZ can be evaluated using Lidar point clouds by utilizing the depth information in the 3D point clouds [53]. Chapter 2 shows that Neural networks can learn complex rules related to LZ identification using labeled training data provided that there is sufficient quantity and diversity in training data representative of operational conditions. There are two main architectures used in

point cloud semantic segmentation, namely, point-based and project-based. As mentioned in chapter 2, a high throughput rate in point cloud semantic segmentation is possible with project-based methods [64].

The main limitation of the available benchmark studies on project-based methods is the point clouds used in the evaluation were generated using a Velodyne 64 Lidar sensor. Therefore, it was essential to evaluate projection-based semantic segmentation methods for the point clouds generated by a VLP-16 Lidar sensor and get a sense of speeds achievable by the on-board computed used by the AI4L-112 project. Therefore, an accuracy and speed evaluation of projection-based methods are needed for VLP-16 Lidars on JAX Kit platforms as a part of the development of NN-based LZ detection method.

This chapter briefly evaluates two state-of-the-art point cloud semantic segmentation models for their performance and inference speed on two hardware platforms. The primary objective is to form the basis for the main development that happens in the next chapter.

4.1 Background

The candidate models for projection-based LZ detection were selected based on their performance on the SemanticKITTI benchmark leaderboard of single scan evaluation and the reproducibility of the code and the results.

4.1.1 RangeNet++

RangeNet++ [63] is one of the methods where 2D projections of LiDAR data are used for point cloud semantic segmentation. The point clouds are converted into a range representation by converting each point to spherical coordinates and then to 2D image coordinates. The projection is computed using Eq. (4.1).

$$\begin{pmatrix} u \\ v \end{pmatrix} = \begin{pmatrix} \frac{1}{2}[1 - \arctan(y, x)\pi^{-1}]W \\ [1 - (\arcsin(zr^{-1}) + f_{up})f^{-1}]H \end{pmatrix}, \tag{4.1}$$

where x, y, z = point coordinates, u, v = image coordinates, H, W = height, width of the desired range image, $f = f_{up} + f_{down}$ = vertical FOV of the sensor, and r = range of each point. In this projection, an image of $HxWx5$ dimensions is generated where the 5 channels are 3D point coordinates (x, y, z), intensity (i) value, and the range index (r).

The semantic segmentation on this range image is carried out by a modified version of Darknet53 [83] backbone architecture as shown in Fig. 4.1. The model also utilizes a faster version of the *k-Nearest Neighbour* (kNN) algorithm to predict semantic labels for the whole point cloud. The authors claim that RangeNet++ can run in its totality online using a single GPU. The advantage of RangeNet++ is its integration into SUMA++ [84] architecture which was developed for semantic-based point cloud registration.

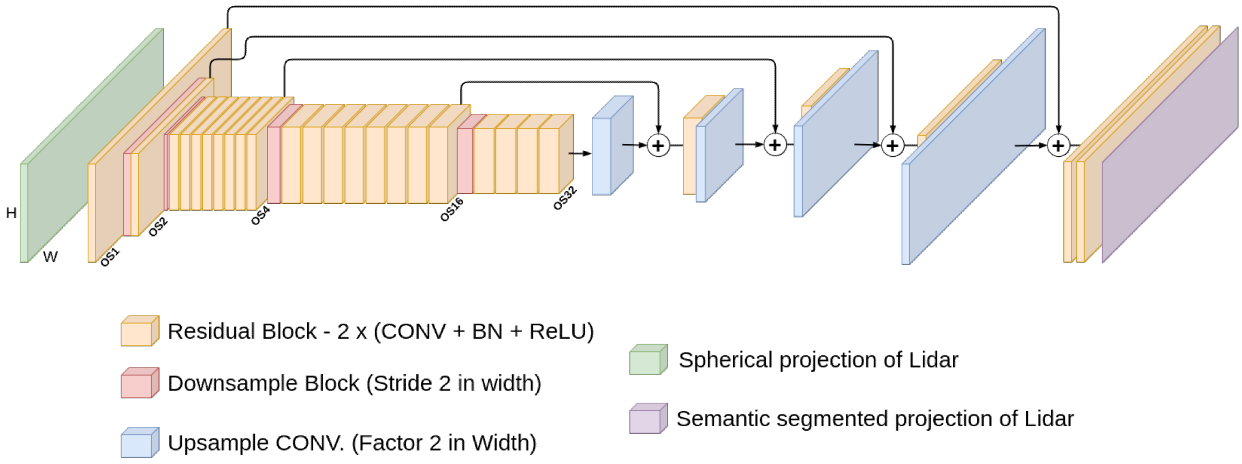


Figure 4.1: Rangenet++ model architecture

4.1.2 SalsaNext

SalsaNext [64] adopts a 2D projection-based architecture similar to RangeNet++, a modified version of its predecessor SalsaNet's [85] encoder-decoder architecture. The 2D spherical projection of the Lidar point cloud is computed using the same equation in Eq. (4.1). The encoder contains a set of ResNet blocks [86] and the decoder applies transpose convolutions and fuses the upsampled features with the residual blocks from the encoder. Figure 4.2 illustrates the SalsaNext architecture, and further in-depth details are available in [64]. According to [85], SalsaNext records a faster runtime than RangeNet++ when evaluated on the same GPU hardware.

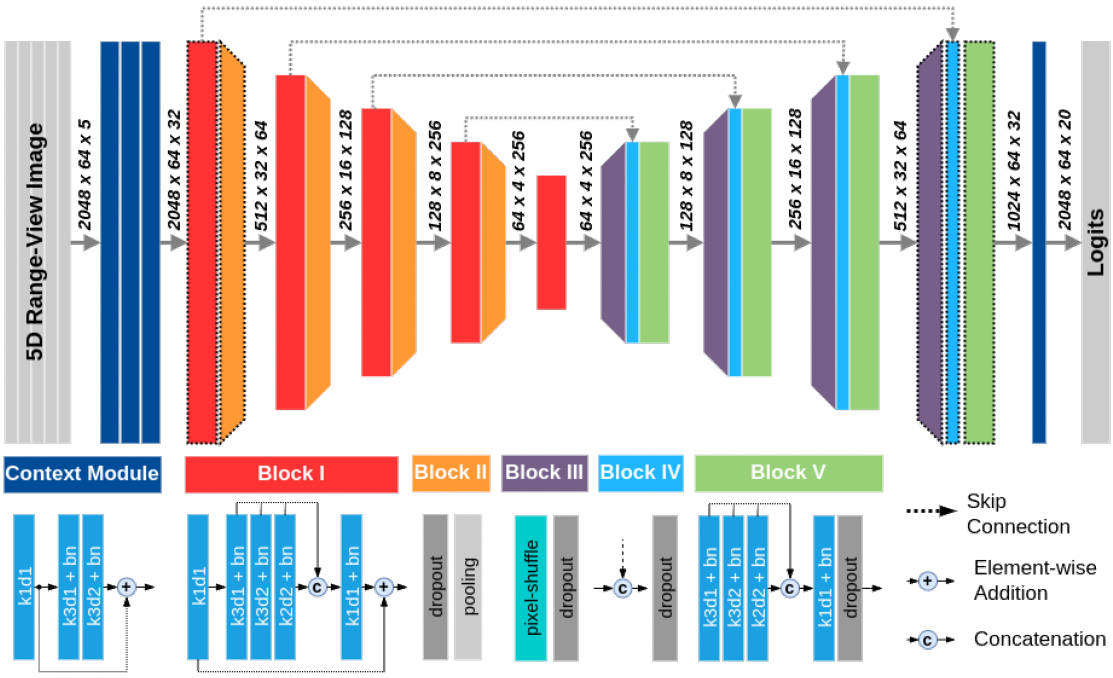


Figure 4.2: SalsaNext model architecture

4.2 Method

Transfer learning was not applied as the main objective of evaluating projection-based semantic segmentation methods is to assess their accuracy-speed compatibility with VLP-16 Lidar sensors. One main characteristic of the projection-based methods is that they directly work on raw Lidar scans rather than point cloud maps. Therefore, the selected two models mentioned in section 4.1 were evaluated on the validation sequence 08 of the KITTI Velodyne dataset [87] and a custom dataset captured using the VILOAM payload.

4.2.1 KITTI Velodyne dataset

The KITTI Velodyne dataset is a benchmark dataset developed by the Karlsruhe Institute of Technology and the Toyota Technological Institute in Chicago to provide new real-world challenges in autonomous navigation systems development. The dataset was captured using a car-mounted multi-sensor system, including four high-resolution video cameras, a Velodyne-64 laser scanner, and a GPS localization system by driving around the city of Karlsruhe. The validation set of the Velodyne laser data of the KITTI dataset was used to evaluate the projection-based semantic segmentation methods. As both RangeNet++ and SalsaNext models were trained on the KITTI Velodyne training set, which contains 64 Lidar channels,

the validation set was downsampled to test for 16 channels. The reason for downsampling the KITTI Velodyne data to 16 channels was to compare the performance of the selected models with the MUN VLP-16 dataset, which also has 16 Lidar channels. Figure 4.3 shows a sample image captured from an instance of the KITTI Velodyne dataset with the semantic labels from the SemanticKITTI benchmark [65].

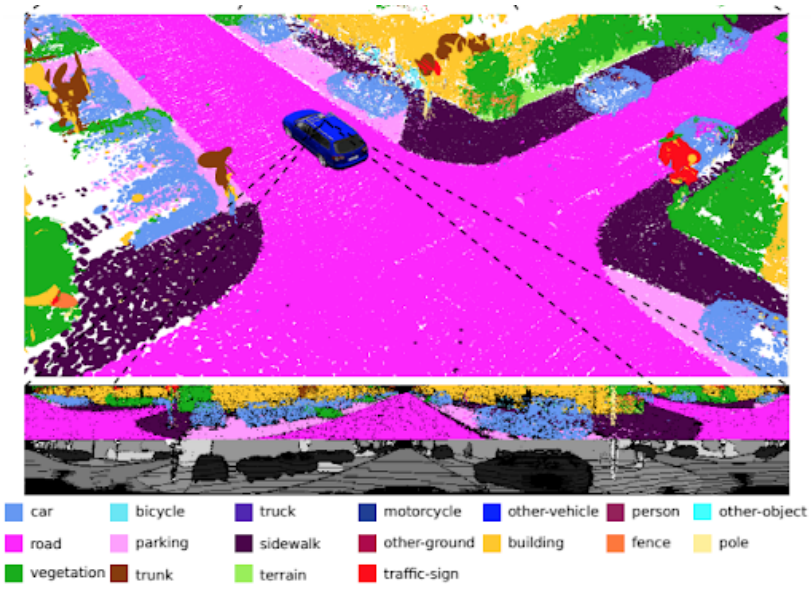


Figure 4.3: Image capture of a sample KITTI Velodyne dataset

4.2.2 Custom MUN VLP-16 dataset

Several datasets were captured using a car-mounted sensors module while driving around the Memorial University of Newfoundland, Canada. The datasets include RGB camera images and laser point clouds captured using a Velodyne VLP-16 rotating 3D laser scanner at 10Hz, GPS measurements, and *Inertial measurement unit* (IMU) data. For the objective of this project, only point cloud data was used. A sample of the GPS traces of one of the datasets is shown in Fig. 4.4.

Figure 4.4: GPS trace of a dataset captured around the university. credits: Didula Dissanayaka

4.3 Qualitative result evaluation

This section provides the quantitative performance evaluation of the projection-based networks on the KITTI Velodyne sequence 08 and the qualitative visual evaluation of the selected MUN VLP-16 data. Since there aren't any ground-truth labels for the MUN VLP-16 data, it is only evaluated for the inference speed using the average runtime and throughput. Predictions are visualized to understand whether the models have performed well by mapping each object class to a color value.

4.3.1 Quantitative Performance Evaluation

Table 4.1 tabulates the performance of the KITTI Velodyne sequence 08 on both selected models. Tables 4.2 and 4.3 show the average runtime of the selected models on each point cloud of the down-sampled (16 channels) KITTI Velodyne sequence 08 and MUN VLP-16 data, respectively.

Table 4.1: Performance of selected models on the KITTI Velodyne Sequence 08 (64 channels)

Model	mIoU(%)		Laptop[1]		Jetson AGX Xavier[2]			
	Recorded[3]	Tested[4]	Avg. runtime(s)	Avg. throughput(pts/s)	Avg. runtime(s)		Avg. throughput(pts/s)[5]	
					15.00 W	MAXN	15.00 W	MAXN
RangeNet ++	52.20	50.30	2.19	55,977.96	1.26	0.66	97,072.09	186,362.37
SalsaNext	59.50	55.80	0.96	127,825.44	0.30	0.16	413,789.41	763,449.39

[1]Laptop with Nvidia Geforce 940MX 4GB GPU

[2]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.

[3]As noted in the respective research articles

[4]As tested on the Laptop and Jetson AGX Xavier

[5]pts/s is points(pts) per second(s)

Table 4.2: Average runtime(in seconds) of selected models on the KITTI Velodyne Sequence 08 (downsampled - 16 channels)

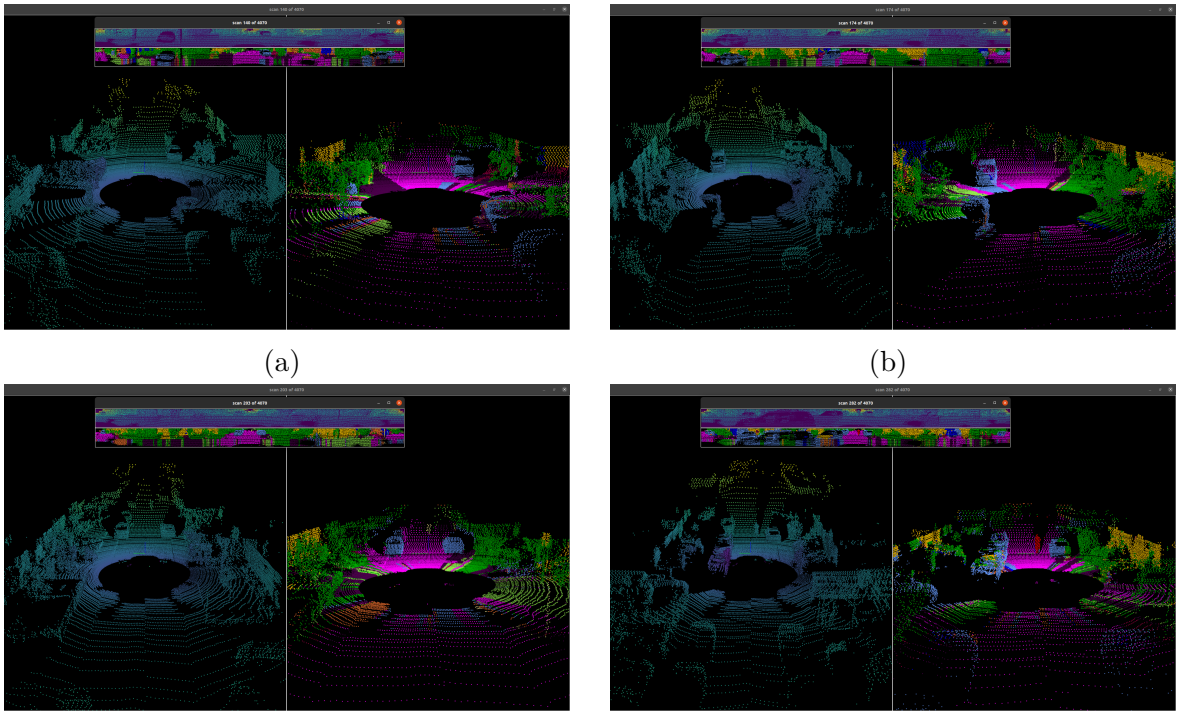
Model	Laptop	Jetson AGX Xavier	
		15 W	MAXN
RangeNet ++	0.425	0.381	0.284
SalsaNext	0.087	0.051	0.034

Table 4.3: Average runtime(in seconds) of selected models on MUN VLP-16 dataset

Model	Laptop	Jetson AGX Xavier	
		15 W	MAXN
RangeNet ++	0.353	0.224	0.145
SalsaNext	0.202	0.052	0.033

4.3.1.1 Qualitative Performance Evaluation

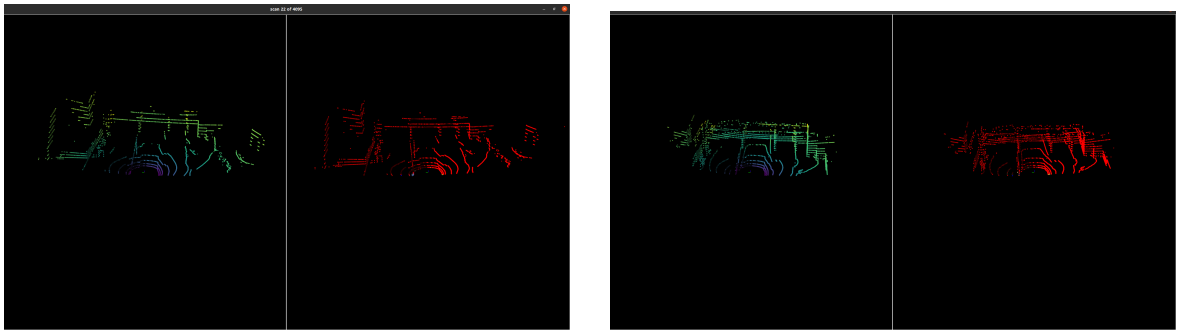
Figures 4.5 and 4.6 show the visualization of the semantic labels predicted by the models on both the down-sampled KITTI Velodyne dataset and MUN VLP-16 data. The semantic predictions for MUN VLP-16 data delivered a low qualitative accuracy, as shown in Fig. 4.6. The poor performance can result from the low point density of VLP-16 Lidars in the MUN VLP-16 data and the difference in the point distribution for each semantic class between the KITTI dataset and the MUN VLP-16 data. This issue can be addressed using a voxel map generated by point cloud accumulation to increase the resolution of the input data. However, literature prefers point-based methods when such map generation is involved, as the generated maps do not have uniform density and have unordered points in the point cloud.



(a) (b)

(c) (d)

Figure 4.5: Visualization of the predictions of the down-sampled KITTI sequence 08 (of each screenshot, left image: colormap of the point cloud, right image: semantic segmentation of the point cloud with Color codes:- pink: road, blue: vehicles, green: vegetation)



(a) (b)

Figure 4.6: Visualization of the predictions of the MUN payload data (of each screenshot, left image: colormap of the point cloud, right image: semantic segmentation of the point cloud).

4.4 Discussion

According to the table 4.1, RangeNet++ and SalsaNext models generated mIoU values far less than 80% for the KITTI Velodyne Sequence 08 though they delivered faster runtime.

This is an inadequate performance from a NN model for real-time safety-critical applications. Additionally, the visual outputs generated by the models for low-resolution Lidar and downsampled Lidar data show that projection-based methods require transfer learning when different Lidar sensors are used. The main reason is that the spherical projection of the point cloud is generated based on the sensor parameters like resolution, the number of Lidar channels, and FOV. This chapter performed a qualitative evaluation of state-of-the-art projection-based methods when applied on VLP-16 Lidars and JAX computing hardware used by the AI4L-112 project. The results highlight the high throughput rate that is possible with the projection-based methods at the expense of considerable degradation in segmentation accuracy due to the lower resolution of raw Lidar scans. Although increasing the resolution is possible through accumulating point clouds to form maps, this does not guarantee the generation of ordered point clouds with uniform density which is a prerequisite for project-based methods. The preparatory evaluation conducted in this chapter assisted with the design decision of moving to point-based methods, which can better handle point cloud maps generated using Lidars such as VLP-16. The next steps in the development process, which use point cloud maps and point-based segmentation methods, are presented in Chapter 5. These circumstances suggest runtime optimization for point-based point cloud semantic segmentation methods as they can deliver higher performance and are independent of sensor parameters.

Chapter 5

LZ detection using point-based methods

As discussed in Chapter 4, although speed improvement is possible with the projection-based network, the accuracy of projection-based methods is still questionable, especially for safety-critical decisions like detecting landing zones. Additionally, these projection-based methods are designed to run on raw data, and point-based methods are developed 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 before range image generation for reasonable results. Also, this won't guarantee the generation of ordered point clouds with the uniform distribution required for projection-based methods, which can result in inaccurate LZ detection. In safety-critical applications, high accuracy and precision in identifying safe LZs are more pertinent than the detection speed. Therefore, this chapter investigates point-based methods for point cloud segmentation and landing zone detection. It will also evaluate the feasibility of integrating a deep neural network-based LZ detection module into a Lidar mapping pipeline and investigate how point cloud aggregation or sub-mapping (which will be described in the section 5.2.1.3) can be utilized to deliver an accuracy-runtime trade-off for online LZ detection.

The first section will describe the selected point-based model and the reason for the selection. The second section will overview the methodology, including the datasets used in transfer learning and testing. The third section will present the point cloud processing module, which is designed to be compatible with the overall architecture and evaluates the capability of the system to provide landing zone proposals in a mapping pipeline. The fourth section of this chapter will illustrate the experimental results and the performance evaluation of

the developed model for the LZ detection method and show how the proposed method can achieve the target accuracy-runtime trade-off when three variations of aerial Lidar datasets are presented. The fourth section will also illustrate sample visualizations of how the NN model achieves the objective of LZ detection in a VILOAM navigation pipeline of a VTOL aircraft. The last section of this chapter will discuss the results, issues related to the proposed method, and the next steps that can be done as future work.

5.1 Model Selection

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 which can free the central processing unit (CPU) for other tasks, ease in fusing different NN networks, and code optimization capability compared to classical methods. The top-performing point-based semantic segmentation model in the Semantic3D leaderboard is evaluated for its performance on three variations of point cloud data to achieve this objective.

5.1.1 ConvPoint Model

As described in the chapter 2, the initial work of point-based point cloud semantic segmentation can be found in [88] and [70]. Due to the ineffectiveness of those two models in terms of runtime-accuracy trade-off and detecting several object classes, the ConvPoint model [73] was selected based on the Semantic3D 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. 5.1, the segmentation network 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 twelve 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 meters wide vertical column centered on this selected point. During testing, the model computes a 2D occupancy pixel map with a "pixel" size of 0.5 meters for 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) which 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.

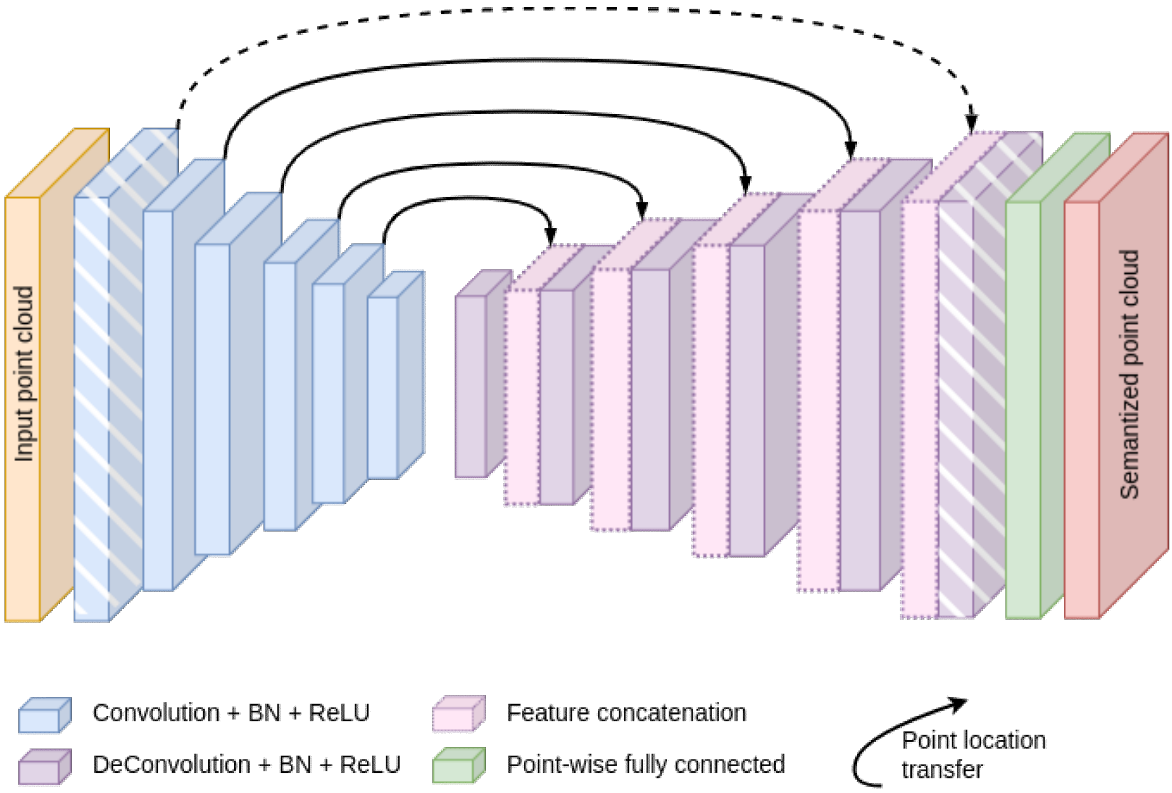


Figure 5.1: Sample illustration of ConvPoint semantic segmentation model

5.2 Methodology

This section will discuss the datasets used for transfer learning and testing for LZ detection, labeling Lidar point clouds for landable and non-landable zones, the transfer learning process, and the point cloud processing module.

5.2.1 Datasets

For this study, several representative datasets should be selected for deep learning module training and testing purposes. Datasets that are freely available online were selected using the following selection criterion. Since this work focuses on determining the LZs of a VTOL, the point clouds should include areas with both landable and non-landable zones for a VTOL drone or a helicopter. As there is a limited number of datasets available to suit the VTOL landing zone application, which captures operational scenarios of VTOL vehicles, e.g., water bodies, infrastructure, parking lots, flat but unlandable surfaces, etc.; in addition to the online datasets, experimental datasets were also captured using a DJI M600 drone (Fig. 5.2)

to support this work. A description of the datasets used for transfer learning is described in this section.

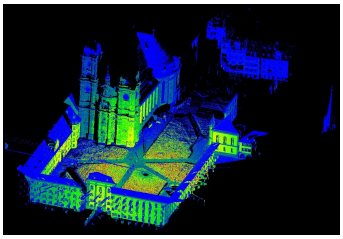
(a) Data collection using MJI M600 quadcopter

(b) LiDAR-drone setup

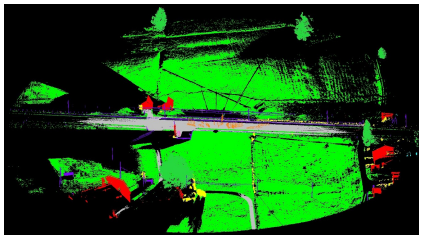
Figure 5.2: Data Collection setup by ISL

5.2.1.1 Semantic3D Benchmark

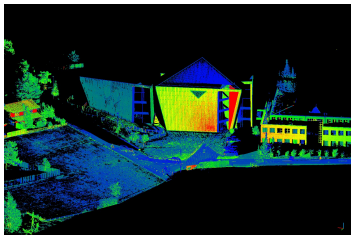
As the online dataset, the semantic3D benchmark [74] was selected. This benchmark has 30 point clouds captured using a survey LiDAR on the ground, which are already categorized into eight classes, (1) man-made terrain, (2) natural terrain, (3) high vegetation, (4) low vegetation, (5) buildings, (6) hard 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. 5.3. The dataset is a good reference to assist with the transfer learning process. For ease in distinguishing between different datasets, this dataset will be called the Semantic3D dataset hereafter in the text.



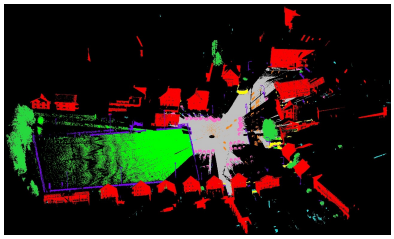
(a) Cathedral1



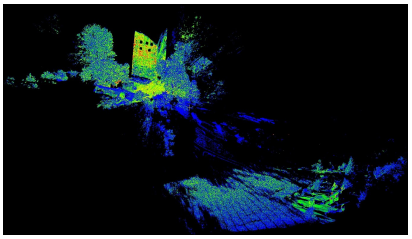
(b) sg27-station1



(c) sg28-station5



(d) sg27-station9

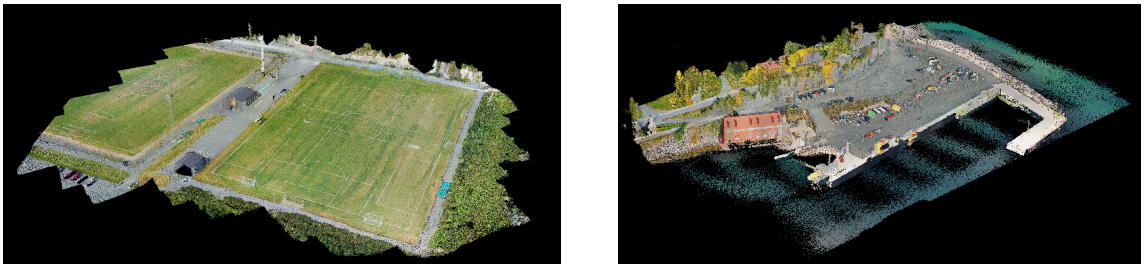


(e) Castle1

Figure 5.3: Semantic3D datasets

5.2.1.2 Experimental data from DJI M600 - MUN dataset 1

Experimental dataset 1 was captured by a Velodyne Lidar sensor attached to a DJI M600 drone over two selected areas, Paradise and Holyrood, in Newfoundland, Canada. The data collected in Paradise, which includes a football ground surrounded by vegetation, is defined as the Paradise dataset. The data captured in Holyrood, which consists of a marine base parking lot, is 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 include flat surfaces like water bodies, frozen lakes, marshlands, unstable ground, etc. Table 5.1 shows the number of points in each dataset and the number of points in the downsampled sets used to test the transfer learned neural network (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. For ease of reference, this dataset will be called the Holyrood-Paradise dataset.



(a) Paradise (b) Holyrood

Figure 5.4: *Intelligent Systems Lab* (ISL) of the *Memorial University of Newfoundland* collected dataset 1

Table 5.1: Number of points in the originally captured dataset vs. downsampled dataset

Dataset	No.of points in original set	No.of points in downsampled set[1]
Paradise	105 005 239	228 673
Holyrood	37 948 660	450 850

[1]The original large point cloud was downsampled using a voxelization method. This approach allows the point clouds to be processed by the selected models within the given memory constraints.

5.2.1.3 Post-processed experimental MUN dataset 1

The proposed architecture described in the section 5.3 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 points. Figure 5.5 shows that the entire point cloud is segmented into 11 sub-clouds using CloudCompare software. Table 5.2 shows the number of points in each segmented point cloud. For ease of reference, this post-processed dataset will be addressed as the post-processed Holyrood dataset.

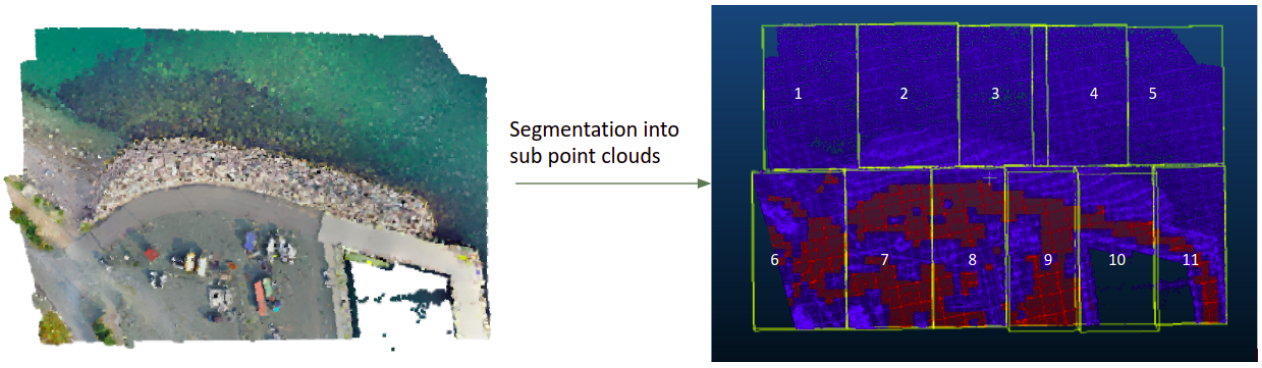


Figure 5.5: Segmentation of Holyrood test set into sub-point clouds

Table 5.2: Number of points in sub-point clouds

Sub cloud	No.of points	Batch sizes[1]	Sampling Sizes[2]
Sub point cloud 1	9065	8, 16, 32, 64	3000, 4000, 5000,.., 9000
Sub point cloud 2	15 793		
Sub point cloud 3	11 632		
Sub point cloud 4	9937		
Sub point cloud 5	8978		
Sub point cloud 6	15 965		
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		

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

The above datasets are generated by combining Lidar scans in a map using point cloud registration software with an offline post-processing workflow. However, these point cloud

maps should be generated during runtime for real-time Landing zone detection. As a result, a suitable Lidar mapping method should be in place. This work uses a VLOAM mapping pipeline developed for the AI4L-112 project for this purpose. The overview of this pipeline and integration of the Landing zone determination module in that pipeline is discussed in the next section 5.3.

5.2.2 Labeling of Training Data

The flow diagram of the LZs labeling process of the dataset is shown in Fig. 5.6. The method used is from [14], 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 $3m \times 3m$ patches, considering the dimensions of the DJIM600 quadcopter and $\pm 1m$ 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 $0.5m$ and maximum z-deviation less than $6m$. 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 $20cm \times 20cm$ region for a $3m \times 3m$ patch [14]. A patch is chosen as landable if its slope is below 5°. These parameters are adjusted to suit each dataset so that representative landing zones are produced as part of the labeling process.

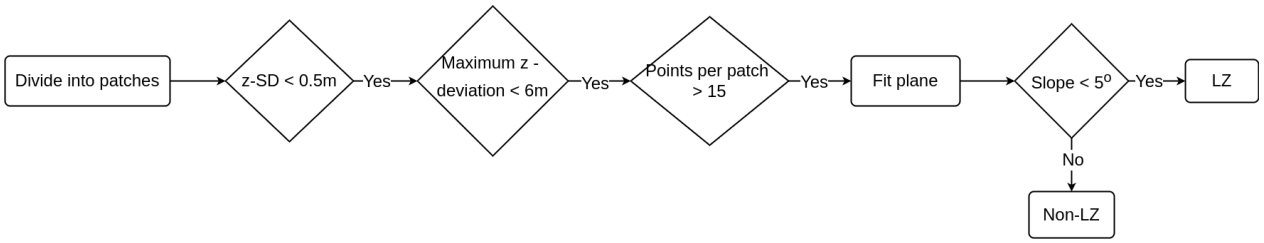
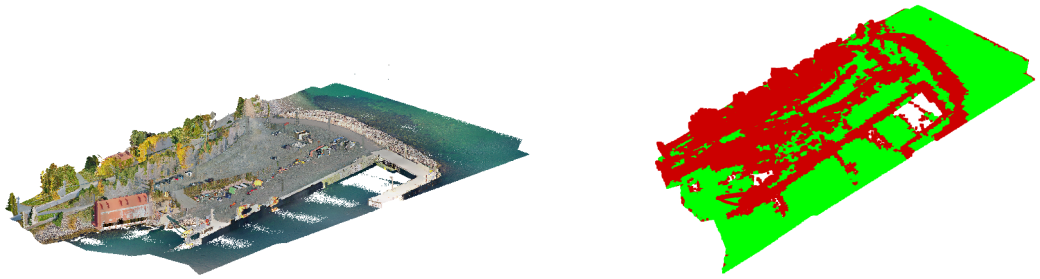


Figure 5.6: Process flow of the geometric labeling

The algorithm in Fig. 5.6 could not detect water bodies, and it wrongly classified water areas as landable zones, as shown in Fig. 5.7. This was resolved using a semantic labeling app by manually selecting water areas and naming the point label as non-landable.



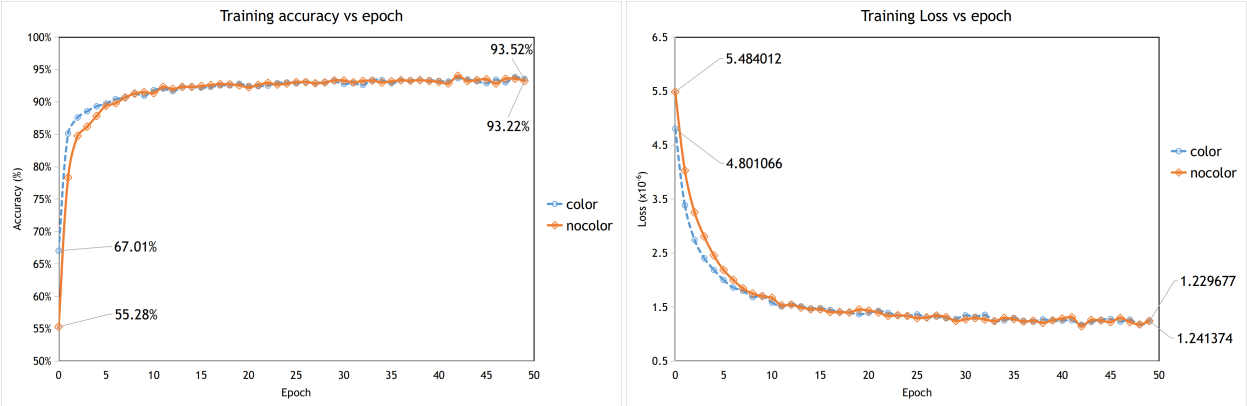
(a) RGB colored Holyrood dataset without outliers

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

Figure 5.7: Original Holyrood dataset and the colored LZs generated by geometric labeling

5.2.3 Transfer Learning

For transfer learning of the ConvPoint model for landing zone identification, the source code was modified to output two object classes (i.e., landable and non-landable) and used the pre-trained weights (pretrained on semantic3D dataset) as the initial checkpoint. As the training set of the transfer learning of ConvPoint, the down-sampled semantic3d dataset and a part of the Holyrood dataset containing points of the water bodies was used. The inputs contained 3D point data, original RGB data, and landable/non-landable labels. The remaining subset of the Holyrood and paradise datasets were used as the testing data. To analyze the effect of RGB values, fine-tuning is done with and without the RGB data input. Figures 5.8(a) and 5.8(b) illustrate the transfer learning accuracy and loss, respectively, 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.



(a) Transfer learning accuracy

(b) Transfer learning loss

Figure 5.8: Transfer learning graphs

5.3 Point cloud processing module

VLOAM [89] 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 six *Degrees 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 sensors system. 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 the proposed architecture, the point cloud semantic segmentation module is embedded into a VLOAM subsystem as shown in Fig. 5.9. In the VLOAM subsystem, there are three main sensor systems, i.e., Camera, 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 further accurate and robust.

The LZ detection subsystem has a few components that will use 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.

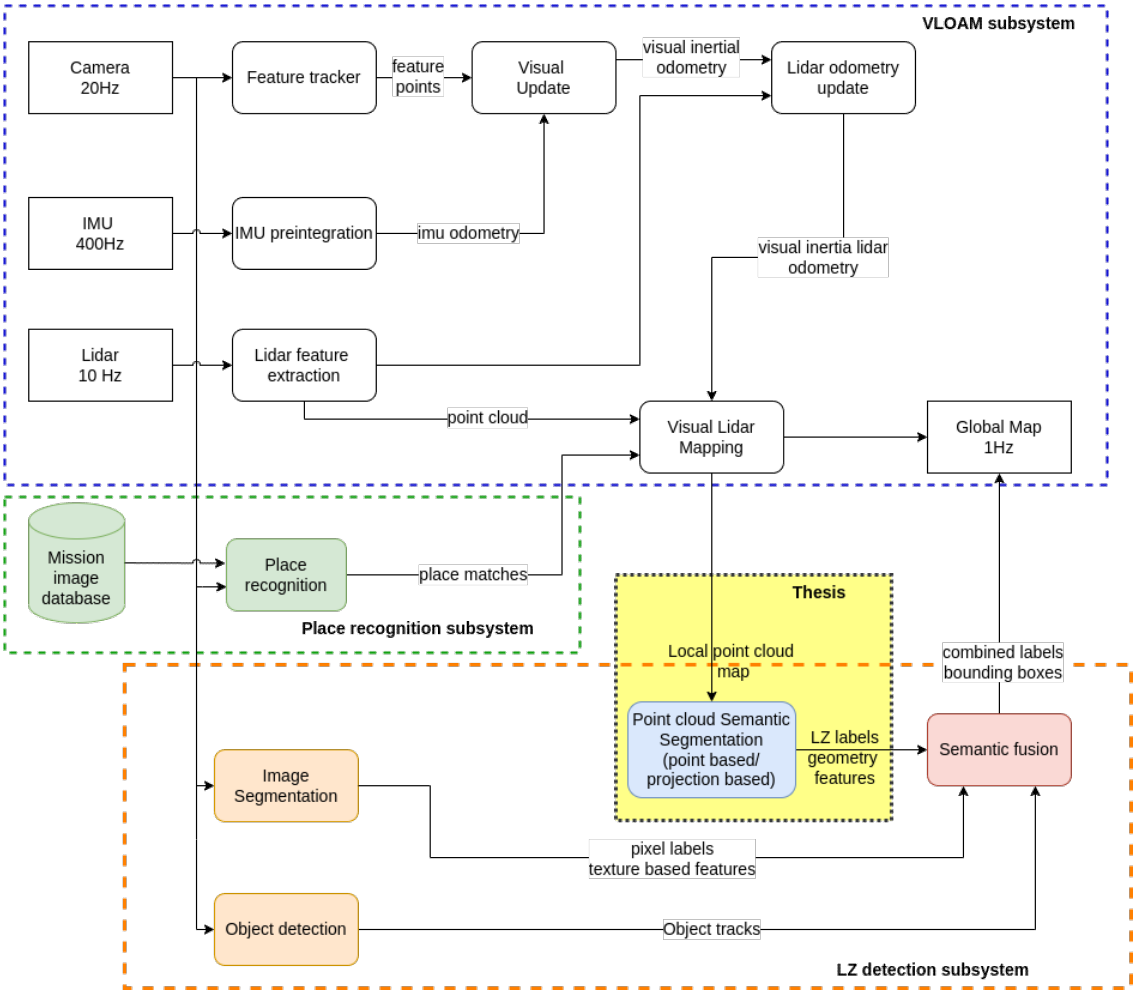


Figure 5.9: Overview of the total architecture and the proposed LZ identification module in this paper (highlighted)

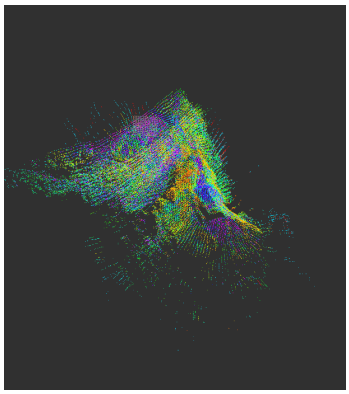
As shown in the above Fig. 5.9, the VLOAM subsystem will generate a local point cloud map with the desired number of points which will then be published into the point cloud semantic segmentation module to generate the landing zone labels. This chapter of the thesis focuses on the LZ labels generated 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 a neural network 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 of the generated point cloud by the VILOAM subsystem for LZ detection purposes.

5.3.1 Datasets generated using VILOAM architecture

This sub-section will overview the aerial Lidar datasets captured using the VLOAM subsystem in Fig. 5.9.

5.3.1.1 Pipeline data - MUN dataset 2

The pipeline data are the local maps generated by the VLOAM module of the proposed system. Figure 5.10(a) shows the point cloud map generated by the VLOAM subsystem, which was implemented on the *Intelligent Systems Lab* (ISL) drone in Fig. 5.10(c), and Fig. 5.10(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 thesis.



(a) Sample point cloud from the MUN dataset 2

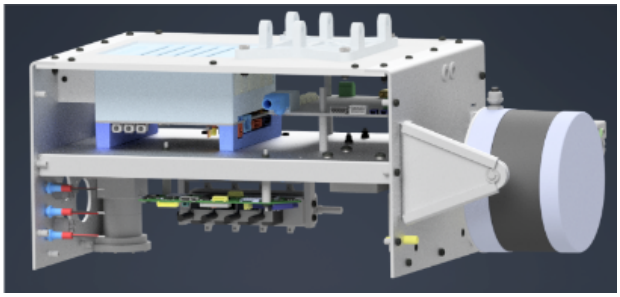
(b) An aerial image of the location

(c) Payload on MUN drone

Figure 5.10: ISL of the *Memorial University of Newfoundland* collected dataset 2

5.3.1.2 Pipeline data - MUN dataset 3

This third dataset was captured using the payload mounted to a Bell-412 helicopter. The payload design is shown in Fi.g 5.11(a), and the Bell-412 helicopter used for taking the data is shown in Fig. 5.11(b). For reference, this dataset will be termed the Bell-412 dataset.



(a) Payload structure

(b) Bell-412 helicopter

Figure 5.11: Bell-412 helicopter setup with the payload

Figure 5.12 illustrates the path of the Bell-412 helicopter on a Google Satellite map based on the GPS data taken by the payload setup. Figure 5.13 is a sample visualization of the map generated by the VLOAM system onboard the payload.

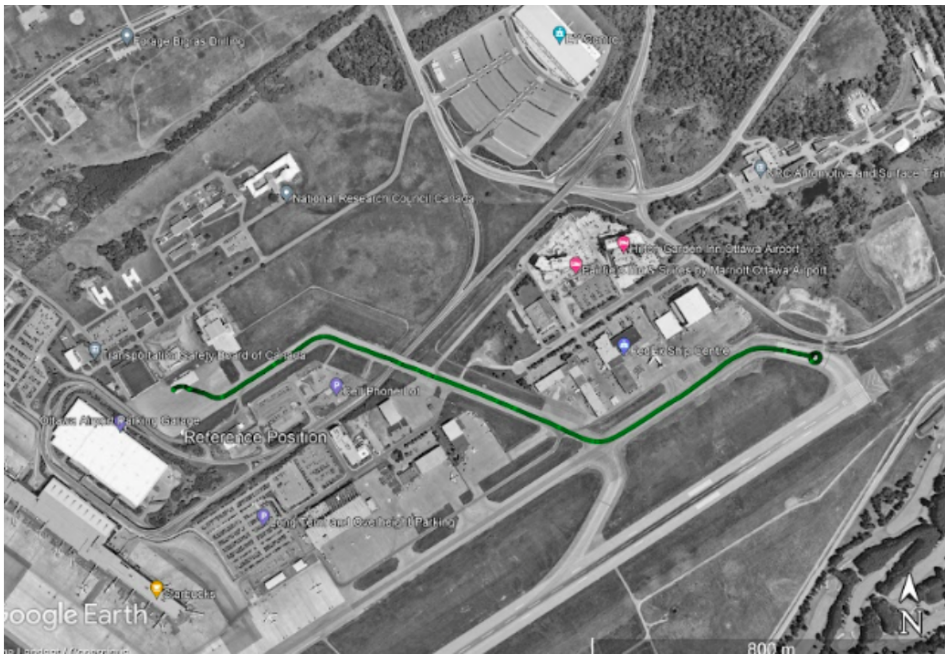


Figure 5.12: Flight path of Bell-412

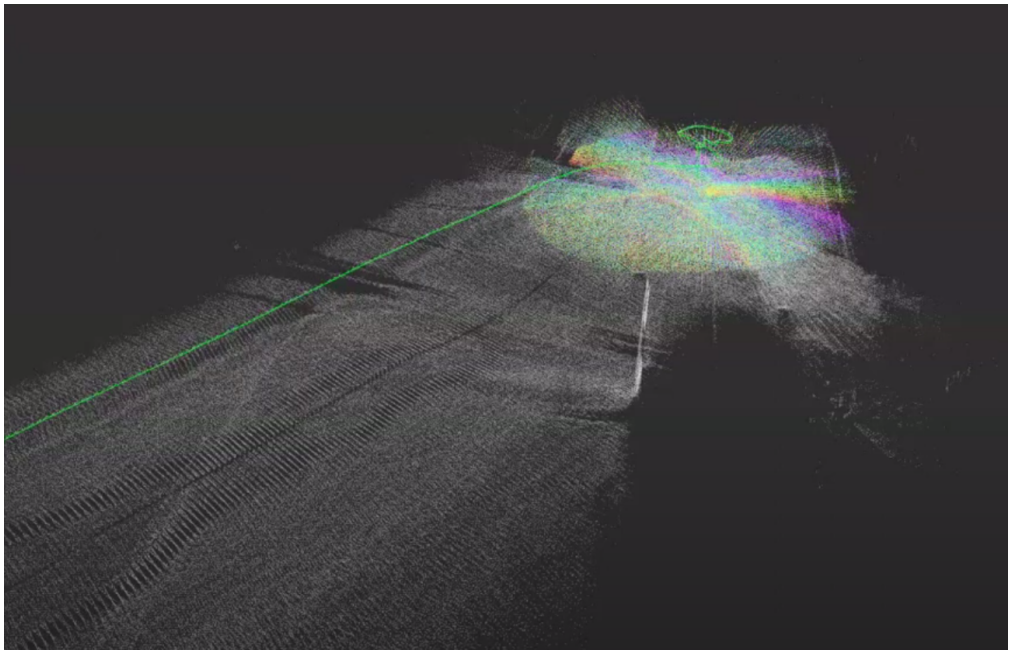


Figure 5.13: Sample Lidar data from the payload onboard Bell-412

5.3.2 Labeling pipeline data

Since there aren't ground truth labels available for the pipeline data, it is required to label the data for landable and non-landable zones. The labeling process in Fig. 5.6 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 should be designed using C++ programming, with the capability of integrating into the VLOAM subsystem. The flow diagram in Fig. 5.14 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.

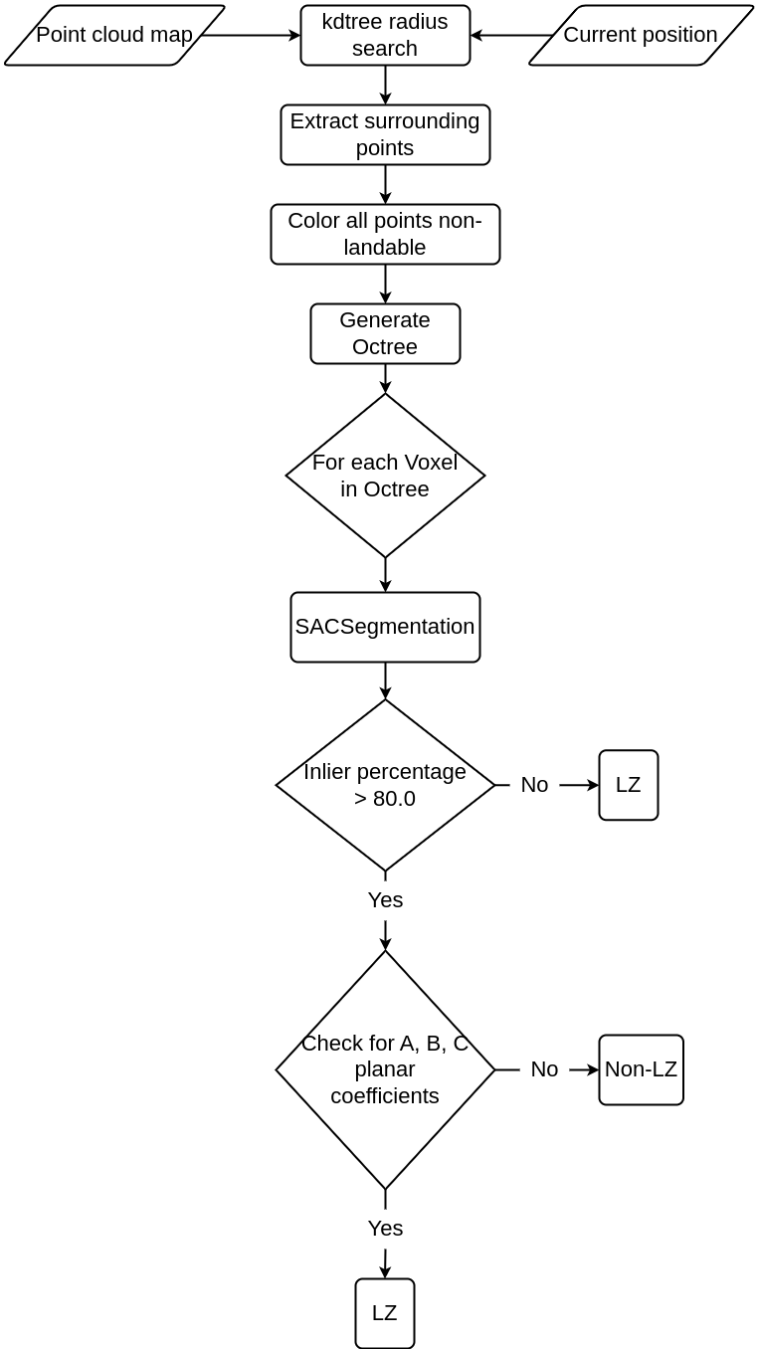


Figure 5.14: Process flow of the labeling of pipeline data

Figures 5.15 and 5.16 show the LZs generated by the algorithm in Fig. 5.14 for two datasets captured by the payload on Bell-412. Further work planned on using this pipeline data on the ConvPoint model will be discussed in section 6.3.

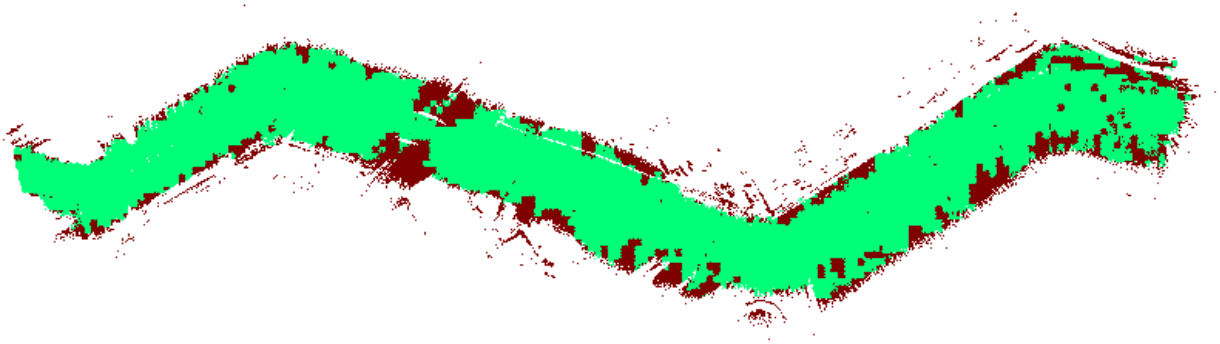


Figure 5.15: Dataset 1 of VLOAM pipeline data captured by Bell-412

Figure 5.16: Dataset 6 of VLOAM pipeline data captured by Bell-412

5.4 Experimental Results

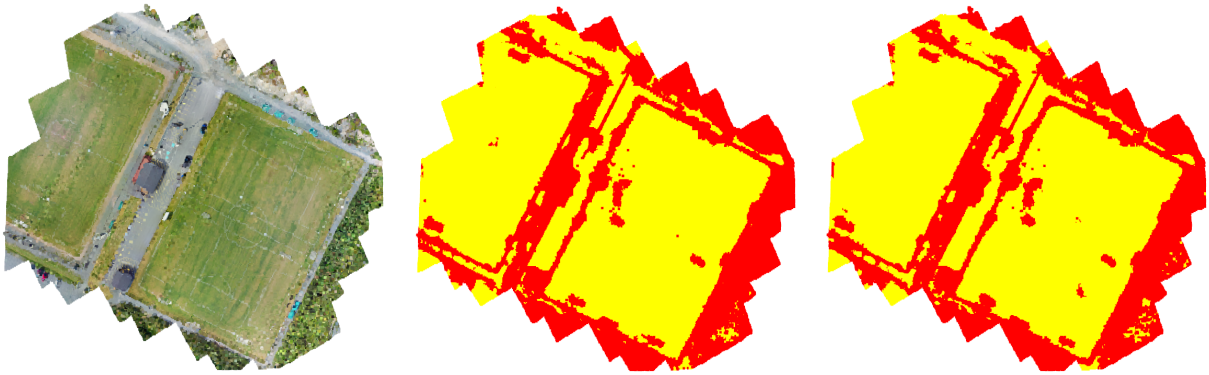
This section provides the main results of the point-based network evaluated on the Holyrood-Paradise dataset, the post-processed Holyrood dataset, the Lighthouse dataset, and the Bell-412 dataset. First, the results of the test data from the Holyrood-Paradise dataset are presented, and then the results on the post-processed Holyrood data will be presented, and

eventually, the results on the Lighthouse dataset 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 LZs 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 use this computing hardware given the 105mm×105mm×65mm form factor, 630g weight, and max 35 Watt power consumption.

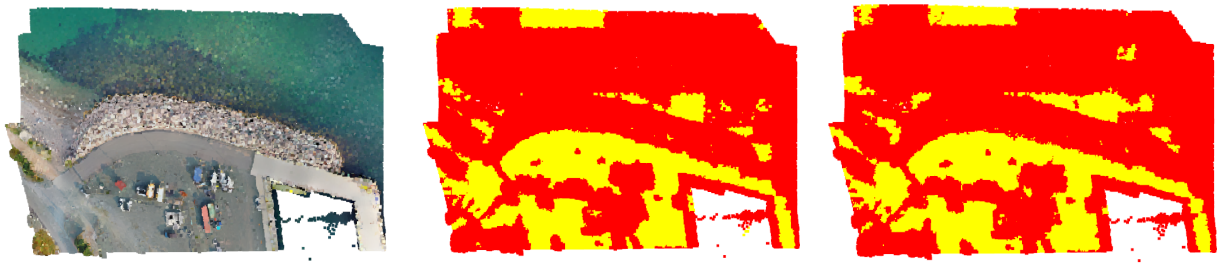
5.4.1 Results: Holyrood-Paradise dataset

The Figs. 5.17,5.18, and Table 5.3 report the quantitative and qualitative performance of the ConvPoint model for LZ detection on the original down-sampled experimental datasets. It is shown that the fine-tuned ConvPoint 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 5.3, the use of maximum power mode 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.



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

Figure 5.17: Visualization of Landing zone detection for Paradise dataset. LZs are in yellow color, and non-LZs are in red color



(a) Holyrood test set (b) Predictions on Google Colab (c) Predictions on Jetson AGX

Figure 5.18: Visualization of Landing zone detection for Holyrood test dataset. LZs are in yellow color, and non-LZs are in red color

Table 5.3: Performance on Holyrood-Paradise dataset

Dataset	Google Colab[1]			Jetson AGX Xavier[2]				
	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	3,940.15	31.25	18.92	7,291.11	12,044.84	89.60
Holyrood	41.63	92.13	3,633.85	29.16	14.78	5,187.30	10,238.37	91.68

[1]Google Colaboratory space equipped with Tesla T4 GPU

[2]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.

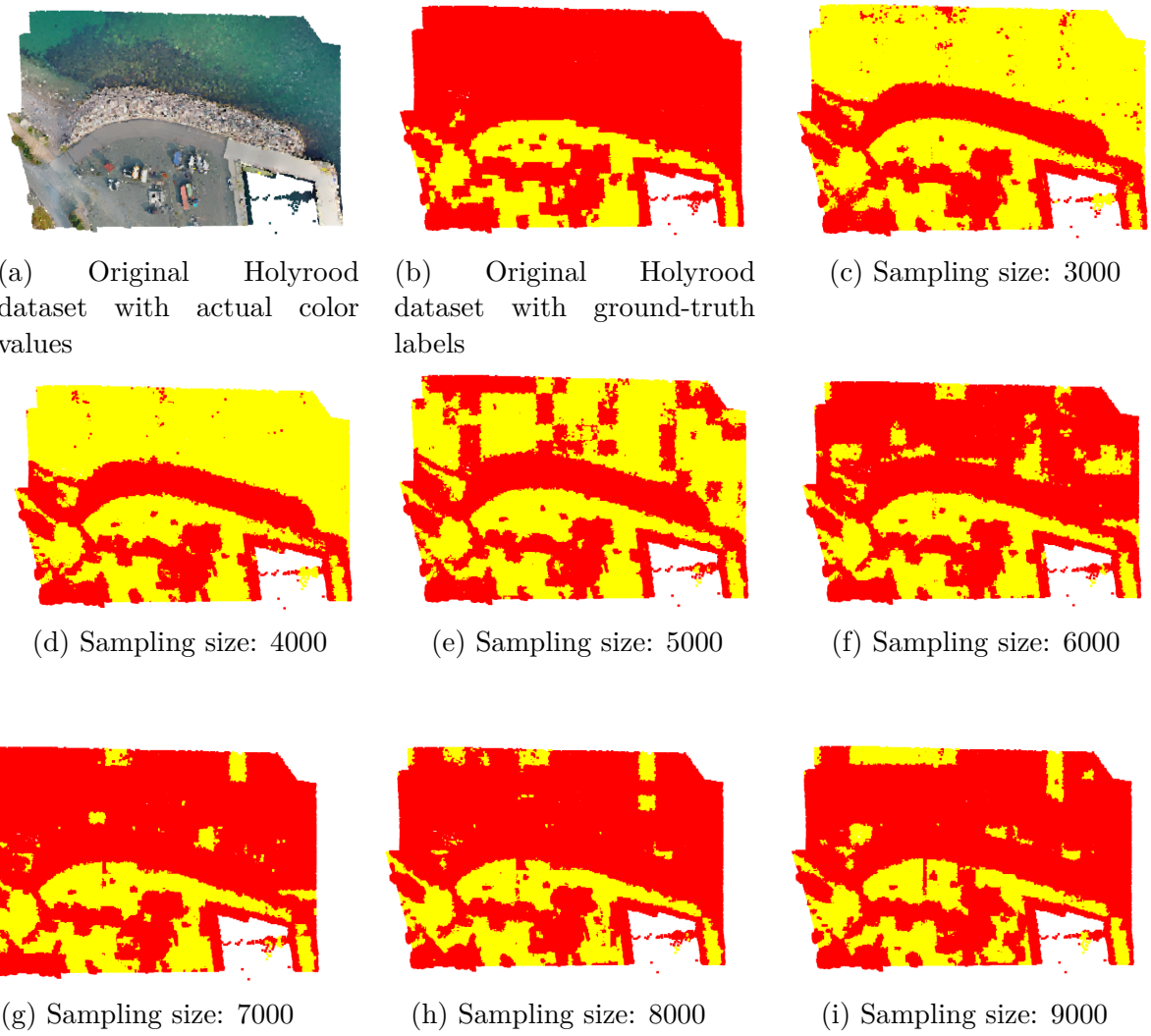
5.4.2 Results: Post-processed Holyrood dataset

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

5.4.2.1 Qualitative Evaluation

Figure 5.19 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 5.19(a) is the original Holyrood test set with the actual color values, while Fig. 5.19(b) contains the color-coded LZs for the original dataset. Figures 5.19(c), 5.19(d), 5.19(e), 5.19(f), 5.19(g), 5.19(h) and 5.19(i) show how the predictions change when the sampling size changes with a given batch size, i.e. 24.



(a) Original Holyrood dataset with actual color values

(b) Original Holyrood dataset with ground-truth labels

(c) Sampling size: 3000

(d) Sampling size: 4000

(e) Sampling size: 5000

(f) Sampling size: 6000

(g) Sampling size: 7000

(h) Sampling size: 8000

(i) Sampling size: 9000

Figure 5.19: Visualization of Landing zone detection for post-processed Holyrood merged point cloud. LZs are in yellow color, and non-LZs are in red color

5.4.2.2 Quantitative Evaluation

Table 5.4 tabulates the quantitative performance of the ConvPoint model for the partitioned Holyrood test set for three different sampling sizes (i.e., 3000, 6000, 9000) with a fixed batch size of 24 on Jetson AGX Xavier when Max power mode was set.

Table 5.4: Performance of ConvPoint model on the post-processed experimental dataset for the sampling sizes 3000, 6000 and 9000

Sub point cloud	Inference times(s)			Throughput(pts/s)			Accuracy (%)		
	3000	6000	9000	3000	6000	9000	3000	6000	9000
sub cloud 1	1.09	1.87	3.01	8,318.40	4,859.32	3,016.23	22.23	94.42	88.94
sub cloud 2	1.10	1.66	2.21	14,392.31	9,532.94	7,135.31	52.88	95.85	92.13
sub cloud 3	0.87	1.33	1.80	13,294.26	8,767.27	6,453.84	44.46	94.11	97.98
sub cloud 4	0.83	1.31	1.78	12,034.65	7,585.73	5,581.70	8.98	72.12	91.94
sub cloud 5	0.72	1.09	1.46	12,513.06	8,223.87	6,167.99	0.70	69.57	99.99
sub cloud 6	0.88	1.34	1.87	18,104.54	11,908.23	8,540.46	88.99	89.68	89.00
sub cloud 7	1.08	1.60	2.43	18,892.94	12,759.38	8,394.01	93.02	92.10	91.91
sub cloud 8	1.12	1.87	2.83	13,761.73	8,228.74	5,420.82	88.21	87.95	83.79
sub cloud 9	0.87	1.34	1.83	17,641.84	11,390.12	8,392.60	94.24	93.67	86.62
sub cloud 10	0.92	1.42	2.02	16,581.09	10,704.60	7,533.99	92.20	94.85	93.30
sub cloud 11	0.89	1.48	2.09	15,323.76	9,210.53	6,506.59	70.65	93.40	95.18

5.4.2.3 Graphical Evaluation

Figures 5.20, 5.21, 5.22 and 5.23 illustrate how the ConvPoint model performed when different combinations of batch sizes and sampling sizes are defined. Figure 5.23 shows how the accuracy would fluctuate for different batch sizes and sampling sizes on the partitioned sets containing water bodies.

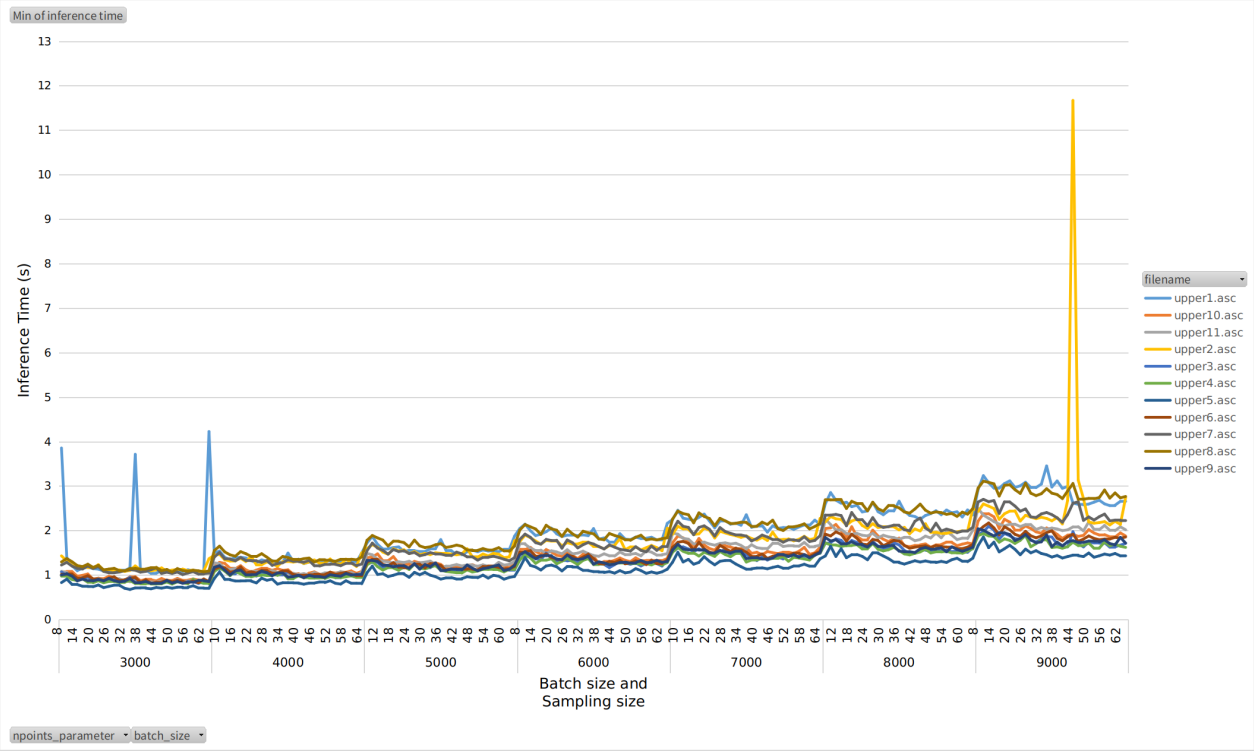


Figure 5.20: Inference time vs. batch size and sampling size for each partitioned dataset

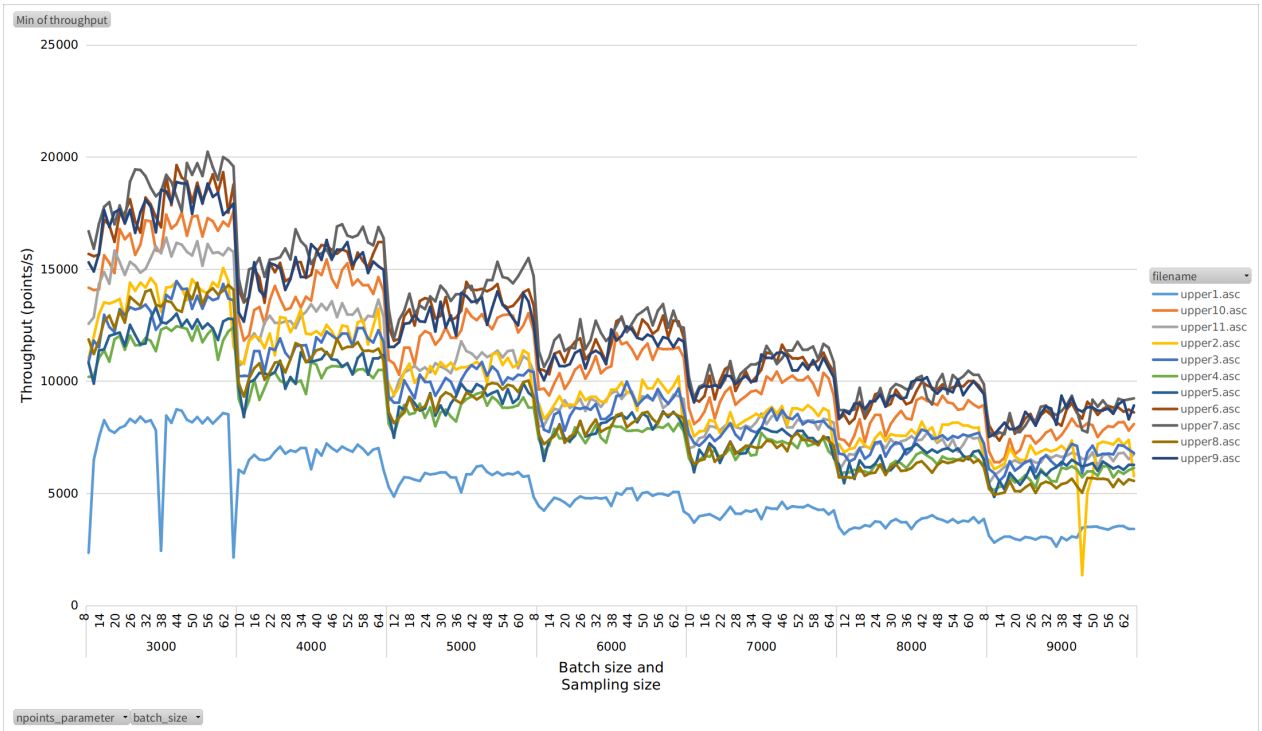


Figure 5.21: Throughput vs. batch size and sampling size for each partitioned dataset

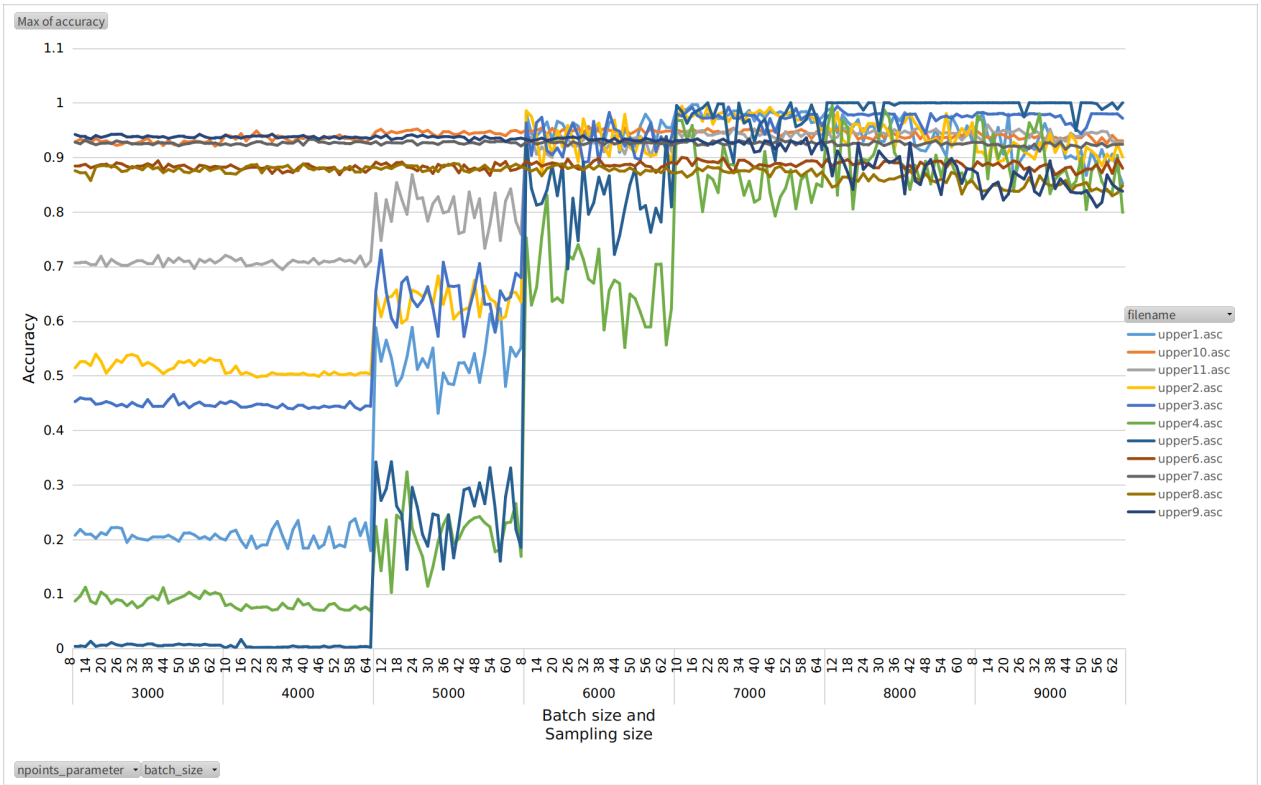


Figure 5.22: Accuracy vs. batch size and sampling size for each partitioned dataset

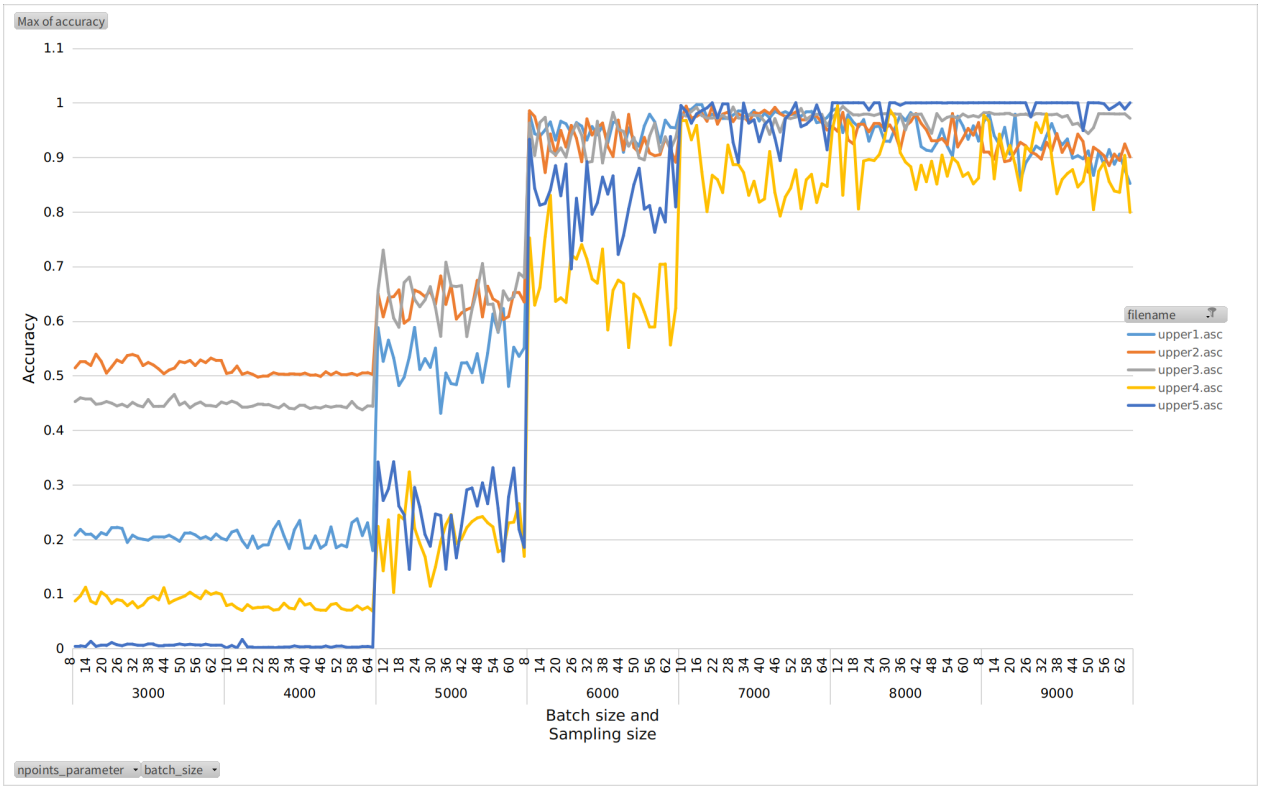


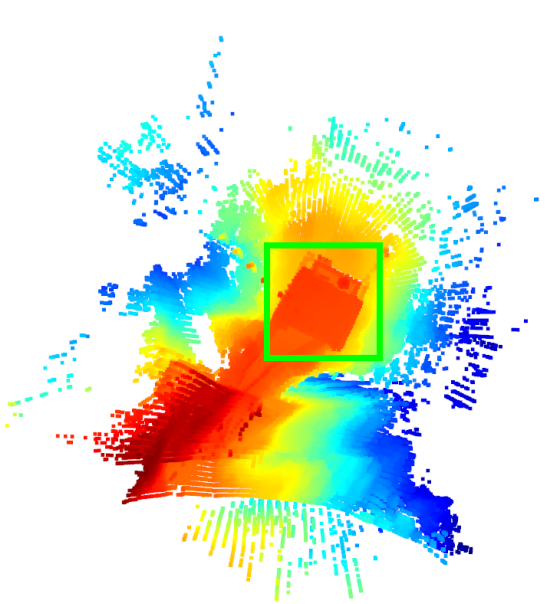
Figure 5.23: Accuracy vs. batch size and sampling size for each partitioned set containing water bodies

Illustrations in sections 5.4.2.1, 5.4.2.2 and 5.4.2.3 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 the Figs. 5.21 and 5.22 it can be perceived that a lower sampling size can generate high throughput. Although smaller sampling sizes have trouble with water bodies detection, it is not the concern of our task as the image segmentation module will handle it. Therefore, a comparatively smaller sampling size with larger batch size and an inference time of 1Hz is a suitable hyperparameter setting to gain a better trade-off for the VLOAM pipeline data rate.

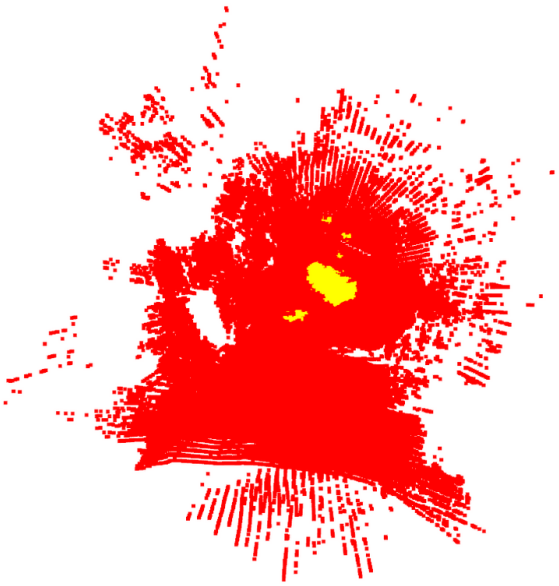
5.4.3 Implementing the LZ detection module on Lighthouse dataset

Figure 5.24 shows LZs predictions generated by the ConvPoint model on a few point cloud maps generated by the VLOAM subsystem. Figure 5.24(a) visualizes the original point cloud maps merged together and the helipad in a green bounded box and Figs. 5.24(b), 5.24(c) and 5.24(d) show how the ConvPoint model detected the helipad as a LZ. From the visual illustrations of the results generated by both the classical method and the ConvPoint NN model, it can be seen that the ConvPoint model has outperformed the classical method in

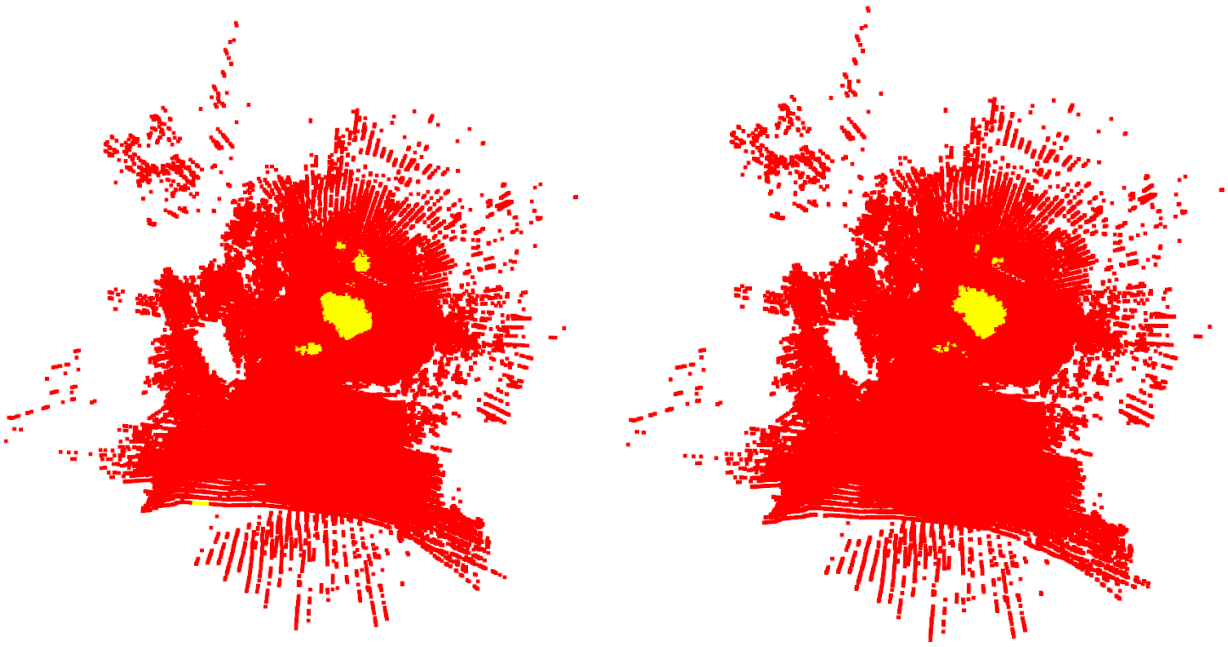
correctly identifying the helipad region as the LZ. The predicted LZ area for the helipad in Fig. 5.24 is smaller than the original helipad shown in the bounding box. The anticipated reason can be the block size value of the infinite column of points used in the transfer learning process of the ConvPoint model or the block size used in the testing process. The runtime for the ConvPoint model to make predictions and the issue with the area of the predicted LZs validate the requirement of further optimization and transfer learning of the NN model.



(a) Merged point cloud maps where the helipad is shown by the green square. A helipad is considered a LZ.



(b) Point cloud map 1



(c) Point cloud map 2 (d) Point cloud map 3

Figure 5.24: Visualization of landing zone detection on several Lighthouse dataset point cloud maps by the ConvPoint model. LZs are in yellow color, and non-LZs are in red color

5.5 Discussion

This report proposes a novel landing zone detection architecture for VTOL vehicles. The developed module can coexist with a VLOAM pipeline of the vehicle and provides a collaborative advantage for the system. Performance evaluation results of a selected point-based semantic segmentation model for landing zone detection are presented on three variations of aerial datasets. When writing this article, the proposed architecture is the best performing approach in safe LZs detection in different landscapes for VTOL vehicles using neural network models compared to the literature discussed in the Introduction section. Based on the Semantic3D benchmark leaderboard, ConvPoint architecture was selected as the target model for the task. The findings showed that the model performs contrastingly for different combinations of hyperparameters, i.e., batch size and sampling size, in terms of the performance metrics, i.e., inference time, throughput, and accuracy. Finding the most suitable hyperparameter combination is essential to derive a better accuracy-runtime trade-off for online LZs detection. Additionally, detecting water bodies, marshlands, and low vegetation as non-landable is crucial for VTOL operations. From the results described in this chapter, it is evident that a larger sampling size should be set to get a comparatively accurate detection

of water areas in the given dataset, 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. The next step of the proposed architecture is the fusion of point cloud labels, pixel labels, and object tracks to deliver combined labels with bounding boxes. 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. Future work on the LZ detection using point cloud semantic segmentation to reach the Lidar update rate will include TensorRT optimizations, further simplification of the NN model, and C++ programming optimization on GPU utilization.

Chapter 6

Conclusion

The main objective of this thesis was to evaluate the performance of deep learning algorithms for intense environment applications. Two studies were discussed in this thesis; sea-ice detection using Deeplab and LZ detection using point cloud semantic segmentation.

6.1 Summary of findings

The objectives of the first study, sea-ice detection, were divided into five deliverable tasks. The first objective was to research and select a state-of-the-art semantic feature-based object detection NN model that can be compared to the baseline NN architecture. Several constraints should be satisfied in selecting such a model, such as object detection using semantic segmentation, transfer-learning ability, the similarity between object classes, and the feasibility of exporting the model into a mobile-compatible format. The PSPNet model was selected as the baseline architecture due to previous work in the same sea-ice detection domain. Based on the criteria, Deeplabv3 was selected for the task of sea-ice detection.

The second and third objectives of the sea-ice detection study were to acquire relevant image data, data preprocessing into the compatible input format, transfer-learn on pretrained weights of Deeplabv3, and validate the performance. Transfer-learning loss and accuracy were calculated for the validation set and then tested for the performance on test data. As the fourth objective, the fine-tuned model was evaluated for its runtime performance and mIoU on a navigation module, and the results were compared with the baseline model. The Deeplabv3 model recorded an IoU of 95.27% for ice class, 98.27% for sky detection, and 91.25% for ocean class. This significantly improved compared to the PSPNet outputs of

94.7%, 95.7%, and 80.8% for the same class labels, respectively. Deeplabv3 took an average of 0.08s per image to generate the semantic labels, which substantially improved inference time compared to 1.91s of the PSPNet model.

The final objective of this study was to export the transfer-learned model into an inference graph that can be deployed on a navigation box and into a lite version for embedding into a mobile application. The exported lite version of the Deeplabv3 was integrated into an application that recorded an inference time of 12s per image.

The challenging component of this study was addressing the lens artifacts that can obstruct the features of the images. This study's future work will consider this issue when generating the training images and labeling the ground truth object classes. Additional future work will include improving the performance of the Deeplabv3 model, the functionality of the model for operational risk assessment, and the detection of different ice categories.

The second study in this thesis is LZ detection using point cloud semantic segmentation. This study is also divided into five objectives based on the problem of LZ detection and expected contributions. The first objective was to research state-of-the-art point cloud semantic segmentation models in projection-based and point-based architectures. Here also, model selection was based on several criteria, including transfer learning capability and similarity between original object classes and new LZ labels.

This study's second and third objectives were to acquire relevant Lidar data for LZ detection, label the datasets into two classes as landable and non-landable using geometric features and manual corrections, transfer-learn the selected models, and validate the performance. The fourth objective was to evaluate the performance of the selected models for the test data and finally test for the accuracy-runtime tradeoff and the feasibility of integrating it in a VLOAM pipeline.

First, projection-based point cloud semantic segmentation methods were evaluated to achieve the LZ detection study's objectives. For this task, RangeNet++ and SalsaNext models were evaluated on the KITTI Velodyne test sequence 08 for their inference runtime and performance in detecting objections. RangeNet++ model delivered a mIoU of 50.30%, an average throughput of 186,362pts/s, and an average runtime of 0.66s, while the SalsaNext model resulted in a mIoU of 55.80%, average throughput of 763,450pts/s and an average runtime of 0.16s for the KITTI Velodyne sequence 08. Though these two models generated a faster

runtime, their mIoU performance was below the expected level for real-time safety-critical applications. Moreover, projection-based networks are designed to run on raw point cloud data, which require point cloud aggregation applied to a low-resolution Lidar, like a VLP-16 sensor, before generating the range image. These complications overruled project-based semantic segmentation methods and demanded a runtime optimization for point-based networks

This study proposed a novel landing zone detection architecture where a point-based point cloud semantic segmentation model will be used with a better runtime-accuracy tradeoff to generate LZ labels and an image-based object detection model to project pixel-based labels onto the point cloud map. The thesis focused on integrating the ConvPoint NN model into the VLOAM pipeline and investigated how point cloud aggregation or sub-mapping can deliver an accuracy-runtime tradeoff for online LZ detection. The ConvPoint model was tested on four variations of aerial Lidar datasets for their performance in terms of mIoU, inference time, and throughput. In the context where ground truth labels were unavailable for the test sets, the LZ predictions of the models were evaluated based on visual outputs.

Finding the optimal hyperparameters set of the ConvPoint model for the inference on Lidar point clouds was the most challenging task. The model was required to predict LZ labels for input point clouds with a rate of 1-10Hz. This was a difficult task as the ConvPoint model is a point-based network and requires more memory and processing time when the point cloud size grew. To address this complication, sub-mapping was attempted, and hyperparameters were tuned accordingly. The thesis graphed the variation of inference time and throughput when different hyperparameter values were set.

Another challenge encountered was the programming language-based optimization. Since the ConvPoint model was coded in Python and PyTorch, there was limited feasibility in converting it into a C++ program. Due to this, publishing the sub-maps generated by the C++ program into a new ROS topic was required, and the Python-based ConvPoint model applied the inference on the sub-maps by subscribing to this topic. The C++-based sub-map generation program could compute at the same speed as the input point cloud maps, but the ConvPoint model could not make the inference work at the same rate. This is a major issue that should be addressed in future work.

6.2 Research Contributions

The contributions of each study can be listed as follows.

- The research findings of the first study: sea-ice detection described in Chapter 3, were published at the OCEANS conference in 2021.
- The results of the initial performance comparison between the classical geometric method and the ConvPoint model for LZ detection on the Holyrood-Paradise dataset were presented at the 77th Annual Forum of the Vertical Flight Society in 2021. The results contained the runtime performance of the ConvPoint model on the Holyrood-Paradise dataset when deployed on the JAX Kit.
- The comparison of projection-based point cloud segmentation methods for VLOAM-based systems was presented at the NECEC 2021 conference.
- The findings and experimental results of integrating the ConvPoint NN model into the VLOAM pipeline and comparing it with the geometric method are in writing for manuscript submission to the Journal of Drone Systems and Applications of Canadian Science Publishing.

6.3 Future Work

The future work of the sea-ice detection component of the thesis will include the classification and detection of different ice types, fine-tuning the NN model for different types of images taken at different locations of the ship, addition of “unclassified" object class to segment entities not belonging to any classes, and fine-tuning the NN model to ignore lens artifacts. The latest development of the Deeplabv3 model, Deeplabv3+, can be evaluated for its performance in sea-ice detection as a part of future work.

The future work of the LZ detection project will consist of further optimizing the NN model for integrating into a VLOAM pipeline using C++ programming, fine-tuning the NN model for different aerial Lidar scenarios, comparing the accuracy-runtime performance between the NN model and the geometric algorithm, and developing the complete LZ detection architecture which can be easily integrated into any VLOAM pipeline. In future work, transfer learning of the ConvPoint model will be carried out using the pipeline data described in section 5.3.2. The findings of the next steps of the LZ detection project will be published at a top-tier conference/ journal.

Bibliography

[1] I. Goodfellow, Y. Bengio, and A. Courville, *Deep learning*. MIT press, 2016.

[2] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[3] D. A. Pomerleau, "Efficient training of artificial neural networks for autonomous navigation," *Neural computation*, vol. 3, no. 1, pp. 88–97, 1991.

[4] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[5] Michon Scott and Kathryn Hansen, *Sea Ice*. [Online]. Available: `https://earthobservatory.nasa.gov/features/SeaIce` (visited on 11/10/2020).

[6] Canadian Coast Guard, "Ice Navigation in Canadian Waters," Tech. Rep., Jul. 2012. [Online]. Available: `https://www.ccg-gcc.gc.ca/publications/icebreaking-deglacage/ice-navigation-glaces/page05-eng.html?wbdisable=true`.

[7] The International Ice Charting Working Group (IICWG), "Ice Information Services: Socio-Economic Benefits and Earth Observation Requirements 2007 Update," The Group on Earth Observation (GEO), Global Monitoring for Environment, and Security (GMES), Tech. Rep., Sep. 2004.

[8] H. Gao, D. Yang, W. Li, Q. Wang, F. Wang, and C. Yin, "Detection of sea ice based on BeiDou-reflected signals," in *International Geoscience and Remote Sensing Symposium (IGARSS)*, vol. 2016-Novem, Beijing, China: Institute of Electrical and Electronics Engineers Inc., Nov. 2016, pp. 4872–4875, ISBN: 9781509033324. DOI: `10.1109/IGARSS.2016.7730271`.

[9] I. Otosaka, M. B. Rivas, and A. Stoffelen, "Bayesian sea ice detection with the ERS scatterometer and sea ice backscatter model at C-band," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 56, no. 4, pp. 2248–2254, Apr. 2018, ISSN: 15580644. DOI: `10.1109/TGRS.2017.2777670`.

[10] M. R. Keller, C. M. Gifford, N. S. Winstead, W. C. Walton, and J. E. Dietz, "Active/Passive Multiple Polarization Sea Ice Detection During Initial Freeze-Up," *IEEE Transactions on Geoscience and Remote Sensing*, pp. 1–15, Aug. 2020, ISSN: 0196-2892. DOI: `10.1109/tgrs.2020.3013512`.

[11] R. Szeliski, *Computer Vision*, ser. Texts in Computer Science. London: Springer London, 2011, ISBN: 978-1-84882-934-3. DOI: `10.1007/978-1-84882-935-0`.

[12] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, P. Martinez-Gonzalez, and J. Garcia-Rodriguez, "A survey on deep learning techniques for image and video semantic segmentation," *Applied Soft Computing Journal*, vol. 70, pp. 41–65, 2018, ISSN: 15684946. DOI: 10.1016/j.asoc.2018.05.018.

[13] B. Dowden, O. De Silva, W. Huang, and D. Oldford, "Sea ice classification via deep neural network semantic segmentation," *IEEE Sensors Journal*, vol. 21, no. 10, pp. 11 879–11 888, 2020.

[14] S. Scherer, L. Chamberlain, and S. Singh, "Autonomous landing at unprepared sites by a full-scale helicopter," *Robotics and Autonomous Systems*, vol. 60, no. 12, pp. 1545–1562, 2012.

[15] D. Maturana and S. Scherer, "3d convolutional neural networks for landing zone detection from lidar," in *2015 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2015, pp. 3471–3478.

[16] N. Otsu, "A threshold selection method from gray level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, pp. 62–66, 1979.

[17] R. Maini and H. Aggarwal, "Study and comparison of various image edge detection techniques," *International journal of image processing (IJIP)*, vol. 3, no. 1, pp. 1–11, 2009.

[18] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[19] Q. Zhang and R. Skjetne, "Image processing for identification of sea-ice floes and the floe size distributions," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 53, no. 5, pp. 2913–2924, May 2015, ISSN: 01962892. DOI: 10.1109/TGRS.2014.2366640.

[20] Q. Zhang, R. Skjetne, I. Metrikin, and S. Løset, "Image processing for ice floe analyses in broken-ice model testing," *Cold Regions Science and Technology*, vol. 111, pp. 27–38, Mar. 2015, ISSN: 0165-232X. DOI: 10.1016/J.COLDREGIONS.2014.12.004.

[21] H. Su, B. Ji, and Y. Wang, "Sea Ice Extent Detection in the Bohai Sea Using Sentinel-3 OLCI Data," *Remote Sensing 2019, Vol. 11, Page 2436*, vol. 11, no. 20, p. 2436, Oct. 2019. DOI: 10.3390/RS11202436.

[22] B. Weissling, S. Ackley, P. Wagner, and H. Xie, "EISCAM — Digital image acquisition and processing for sea ice parameters from ships," *Cold Regions Science and Technology*, vol. 57, no. 1, pp. 49–60, Jun. 2009, ISSN: 0165-232X. DOI: 10.1016/J.COLDREGIONS.2009.01.001.

[23] H.-M. Heyn, M. Knoche, Q. Zhang, and R. Skjetne, "A System for Automated Vision-Based Sea-Ice Concentration Detection and Floe-Size Distribution Indication From an Icebreaker," *Proceedings of the International Conference on Offshore Mechanics and Arctic Engineering - OMAE*, vol. 8, Sep. 2017. DOI: 10.1115/OMAE2017-61822.

[24] A. Sandru, H. Hyyti, A. Visala, and P. Kujala, "A Complete Process For Shipborne Sea-Ice Field Analysis Using Machine Vision," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 14 539–14 545, Jan. 2020, ISSN: 2405-8963. DOI: 10.1016/J.IFACOL.2020.12.1458.

[25] M. Everingham, S. M. A. Eslami, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes challenge: A retrospective," *International Journal of Computer Vision*, vol. 111, no. 1, pp. 98–136, Jan. 2015.

[26] T.-Y. Lin, M. Maire, S. Belongie, *et al.*, *Microsoft coco: Common objects in context*, 2014. DOI: 10.48550/ARXIV.1405.0312. [Online]. Available: https://arxiv.org/abs/1405.0312.

[27] M. Cordts, M. Omran, S. Ramos, *et al.*, "The cityscapes dataset for semantic urban scene understanding," in *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[28] J. Long, E. Shelhamer, and T. Darrell, "Fully convolutional networks for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3431–3440.

[29] O. Ronneberger, P.Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, ser. LNCS, (available on arXiv:1505.04597 [cs.CV]), vol. 9351, Springer, 2015, pp. 234–241.

[30] A. S. Nagi, M. S. Minhas, L. Xu, and K. A. Scott, "A Multi-Scale Technique to Detect Marginal Ice Zones Using Convolutional Neural Networks," *International Geoscience and Remote Sensing Symposium (IGARSS)*, pp. 3035–3038, Sep. 2020. DOI: 10.1109/IGARSS39084.2020.9324172.

[31] A. S. Nagi, D. Kumar, D. Sola, and K. A. Scott, "RUF: Effective Sea Ice Floe Segmentation Using End-to-End RES-UNET-CRF with Dual Loss," *Remote Sensing 2021, Vol. 13, Page 2460*, vol. 13, no. 13, p. 2460, Jun. 2021. DOI: 10.3390/RS13132460.

[32] Y. Han, Y. Liu, Z. Hong, Y. Zhang, S. Yang, and J. Wang, "Sea Ice Image Classification Based on Heterogeneous Data Fusion and Deep Learning," *Remote Sensing 2021, Vol. 13, Page 592*, vol. 13, no. 4, p. 592, Feb. 2021. DOI: 10.3390/RS13040592.

[33] X. Zhang, J. Jin, Z. Lan, *et al.*, "Icenet: A semantic segmentation deep network for river ice by fusing positional and channel-wise attentive features," *Remote Sensing*, vol. 12, no. 2, 2020, ISSN: 2072-4292. DOI: 10.3390/rs12020221.

[34] E. Kim, G. S. Dahiya, S. Løset, and R. Skjetne, "Can a computer see what an ice expert sees? Multilabel ice objects classification with convolutional neural networks," *Results in Engineering*, vol. 4, p. 100 036, Dec. 2019, ISSN: 25901230. DOI: 10.1016/j.rineng.2019.100036.

[35] O.-M. Pedersen and E. Kim, "Arctic Vision: Using Neural Networks for Ice Object Classification, and Controlling How They Fail," *Journal of Marine Science and Engineering 2020, Vol. 8, Page 770*, vol. 8, no. 10, p. 770, Sep. 2020. DOI: 10.3390/JMSE8100770.

[36] E. Kim, N. Panchi, and G. S. Dahiya, "Towards automated identification of ice features for surface vessels using deep learning," *Journal of Physics: Conference Series*, vol. 1357, no. 1, p. 012 042, Oct. 2019, ISSN: 1742-6596. DOI: 10.1088/1742-6596/1357/1/012042.

[37] O. Russakovsky, J. Deng, H. Su, *et al.*, "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015. DOI: 10.1007/s11263-015-0816-y.

[38] S. Sorensen, V. Veerendraveer, W. Treible, A. R. Mahoney, and C. Kambhamettu, "The polar sea ice topography reconstruction system," *Annals of Glaciology*, vol. 61, no. 82, pp. 127–138, Sep. 2020, ISSN: 0260-3055. DOI: 10.1017/AOG.2020.21.

[39] R. Qin, J. Tian, and P. Reinartz, "3d change detection–approaches and applications," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 122, pp. 41–56, 2016.

[40] A. Nguyen and B. Le, "3d point cloud segmentation: A survey," in *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*, IEEE, 2013, pp. 225–230.

[41] T. Rabbani, F. Van Den Heuvel, and G. Vosselmann, "Segmentation of point clouds using smoothness constraint," *International archives of photogrammetry, remote sensing and spatial information sciences*, vol. 36, no. 5, pp. 248–253, 2006.

[42] B. Bhanu, S. Lee, C.-C. Ho, and T. Henderson, "Range data processing: Representation of surfaces by edges," in *Proceedings of the eighth international conference on pattern recognition*, IEEE Computer Society Press, 1986, pp. 236–238.

[43] X. Y. Jiang, U. Meier, and H. Bunke, "Fast range image segmentation using high-level segmentation primitives," in *Proceedings Third IEEE Workshop on Applications of Computer Vision. WACV'96*, IEEE, 1996, pp. 83–88.

[44] Y. Xie, J. Tian, and X. X. Zhu, "Linking points with labels in 3d: A review of point cloud semantic segmentation," *IEEE Geoscience and Remote Sensing Magazine*, vol. 8, no. 4, pp. 38–59, 2020.

[45] J.-E. Deschaud and F. Goulette, "A fast and accurate plane detection algorithm for large noisy point clouds using filtered normals and voxel growing," in *3DPVT*, Hal Archives-Ouvertes Paris, France, 2010.

[46] A.-V. Vo, L. Truong-Hong, D. F. Laefer, and M. Bertolotto, "Octree-based region growing for point cloud segmentation," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 104, pp. 88–100, 2015.

[47] M. A. Fischler and R. C. Bolles, "Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography," *Communications of the ACM*, vol. 24, no. 6, pp. 381–395, 1981.

[48] F. Tarsha-Kurdi, T. Landes, and P. Grussenmeyer, "Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data," in *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, vol. 36, 2007, pp. 407–412.

[49] A. E. Johnson, A. R. Klumpp, J. B. Collier, and A. A. Wolf, "Lidar-based hazard avoidance for safe landing on mars," *Journal of guidance, control, and dynamics*, vol. 25, no. 6, pp. 1091–1099, 2002.

[50] M. Garg, A. Kumar, and P. B. Sujit, "Terrain-based landing site selection and path planning for fixed-wing UAVs," *2015 International Conference on Unmanned Aircraft Systems, ICUAS 2015*, pp. 246–251, Jul. 2015. DOI: 10.1109/ICUAS.2015.7152297.

[51] O. G. Lorenzo, J. Martínez, D. L. Vilariño, T. F. Pena, J. C. Cabaleiro, and F. F. Rivera, "Landing sites detection using LiDAR data on manycore systems," *The Journal of Supercomputing 2016 73:1*, vol. 73, no. 1, pp. 557–575, Nov. 2016, ISSN: 1573-0484. DOI: 10.1007/S11227-016-1912-7. [Online]. Available: https://link.springer.com/article/10.1007/s11227-016-1912-7.

[52] G. Loureiro, A. Dias, A. Martins, and J. Almeida, "Emergency landing spot detection algorithm for unmanned aerial vehicles," *Remote Sensing*, vol. 13, no. 10, p. 1930, 2021.

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

[54] M. F. R. Lee, A. Nugroho, T. T. Le, Bahrudin, and S. N. Bastida, "Landing area recognition using deep learning for unmanned aerial vehicles," *International Conference on Advanced Robotics and Intelligent Systems, ARIS*, vol. 2020-Augus, Aug. 2020, ISSN: 25726919. DOI: 10.1109/ARIS50834.2020.9205793.

[55] R. Polvara, S. Sharma, J. Wan, A. Manning, and R. Sutton, "Autonomous Vehicular Landings on the Deck of an Unmanned Surface Vehicle using Deep Reinforcement Learning," *Robotica*, vol. 37, no. 11, pp. 1867–1882, Nov. 2019, ISSN: 0263-5747. DOI: 10.1017/S0263574719000316. [Online]. Available: https://www.cambridge.org/core/journals/robotica/article/abs/autonomous-vehicular-landings-on-the-deck-of-an-unmanned-surface-vehicle-using-deep-reinforcement-learning/6B87C450D4D431EC163DBEA45FD60C73.

[56] S. Lee and Y. Kwon, "Safe landing of drone using ai-based obstacle avoidance," *International Journal of Mechanical Engineering and Robotics Research*, vol. 9, no. 11, 2020.

[57] F. J. Lawin, M. Danelljan, P. Tosteberg, G. Bhat, F. S. Khan, and M. Felsberg, "Deep projective 3d semantic segmentation," in *International Conference on Computer Analysis of Images and Patterns*, Springer, 2017, pp. 95–107.

[58] A. Boulch, B. Le Saux, and N. Audebert, "Unstructured point cloud semantic labeling using deep segmentation networks.," *3dor@ eurographics*, vol. 3, pp. 1–8, 2017.

[59] M. Tatarchenko, J. Park, V. Koltun, and Q.-Y. Zhou, "Tangent convolutions for dense prediction in 3d," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 3887–3896.

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

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

[62] C. Xu, B. Wu, Z. Wang, *et al.*, "Squeezesegv3: Spatially-adaptive convolution for efficient point-cloud segmentation," in *European Conference on Computer Vision*, Springer, 2020, pp. 1–19.

[63] A. Milioto, I. Vizzo, J. Behley, and C. Stachniss, "RangeNet++: Fast and Accurate LiDAR Semantic Segmentation," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[64] T. Cortinhal, G. Tzelepis, and E. E. Aksoy, *Salsanext: Fast, uncertainty-aware semantic segmentation of lidar point clouds for autonomous driving*, 2020. arXiv: 2003.03653 [cs.CV].

[65] J. Behley, M. Garbade, A. Milioto, *et al.*, "SemanticKITTI: A Dataset for Semantic Scene Understanding of LiDAR Sequences," in *Proc. of the IEEE/CVF International Conf. on Computer Vision (ICCV)*, 2019.

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

[67] X. Ye, J. Li, H. Huang, L. Du, and X. Zhang, "3d recurrent neural networks with context fusion for point cloud semantic segmentation," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 403–417.

[68] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[69] Z. Wu, S. Song, A. Khosla, *et al.*, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[70] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *CoRR*, vol. abs/1706.02413, 2017. arXiv: 1706.02413. [Online]. Available: http://arxiv.org/abs/1706.02413.

[71] Q. Hu, B. Yang, L. Xie, *et al.*, "Learning semantic segmentation of large-scale point clouds with random sampling," *IEEE Transactions on Pattern Analysis & Machine Intelligence*, no. 01, pp. 1–1, May 2021, ISSN: 1939-3539. DOI: 10.1109/TPAMI.2021.3083288.

[72] H. Thomas, C. R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, and L. J. Guibas, "Kpconv: Flexible and deformable convolution for point clouds," *Proceedings of the IEEE International Conference on Computer Vision*, 2019.

[73] A. Boulch, "Convpoint: Continuous convolutions for point cloud processing," *Computers & Graphics*, vol. 88, pp. 24–34, 2020.

[74] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "SEMANTIC3D.NET: A new large-scale point cloud classification benchmark," in *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. IV-1-W1, 2017, pp. 91–98.

[75] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking Atrous Convolution for Semantic Image Segmentation," Jun. 2017. arXiv: 1706.05587.

[76] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, *The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results*, http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[77] F. Yu, V. Koltun, and T. Funkhouser, *Dilated residual networks*, 2017. DOI: 10.48550/ARXIV.1705.09914. [Online]. Available: https://arxiv.org/abs/1705.09914.

[78] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia, "Pyramid scene parsing network," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2881–2890.

[79] L. C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 4, pp. 834–848, Apr. 2018, ISSN: 01628828. DOI: 10.1109/TPAMI.2017.2699184. arXiv: 1606.00915.

[80] Y. Chen, T. Yang, X. Zhang, G. Meng, X. Xiao, and J. Sun, "Detnas: Backbone search for object detection," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[81] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L. C. Chen, "MobileNetV2: Inverted Residuals and Linear Bottlenecks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2018, pp. 4510–4520, ISBN: 9781538664209. DOI: 10.1109/CVPR.2018.00474. arXiv: 1801.04381.

[82] A. Bréhéret, *Pixel Annotation Tool*, 2017. [Online]. Available: https://github.com/abreheret/PixelAnnotationTool.

[83] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, *Yolov4: Optimal speed and accuracy of object detection*, 2020. arXiv: 2004.10934 [cs.CV].

[84] X. Chen, A. Milioto, E. Palazzolo, P. Giguère, J. Behley, and C. Stachniss, "SuMa++: Efficient LiDAR-based Semantic SLAM," in *Proceedings of the IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*, 2019.

[85] E. E. Aksoy, S. Baci, and S. Cavdar, "Salsanet: Fast road and vehicle segmentation in lidar point clouds for autonomous driving," in *IEEE Intelligent Vehicles Symposium (IV2020)*, 2020.

[86] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2016-Decem, IEEE Computer Society, Dec. 2016, pp. 770–778, ISBN: 9781467388504. DOI: 10.1109/CVPR.2016.90. arXiv: 1512.03385.

[87] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2012.

[88] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[89] J. Zhang and S. Singh, "Visual-lidar odometry and mapping: Low-drift, robust, and fast," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2015, pp. 2174–2181.