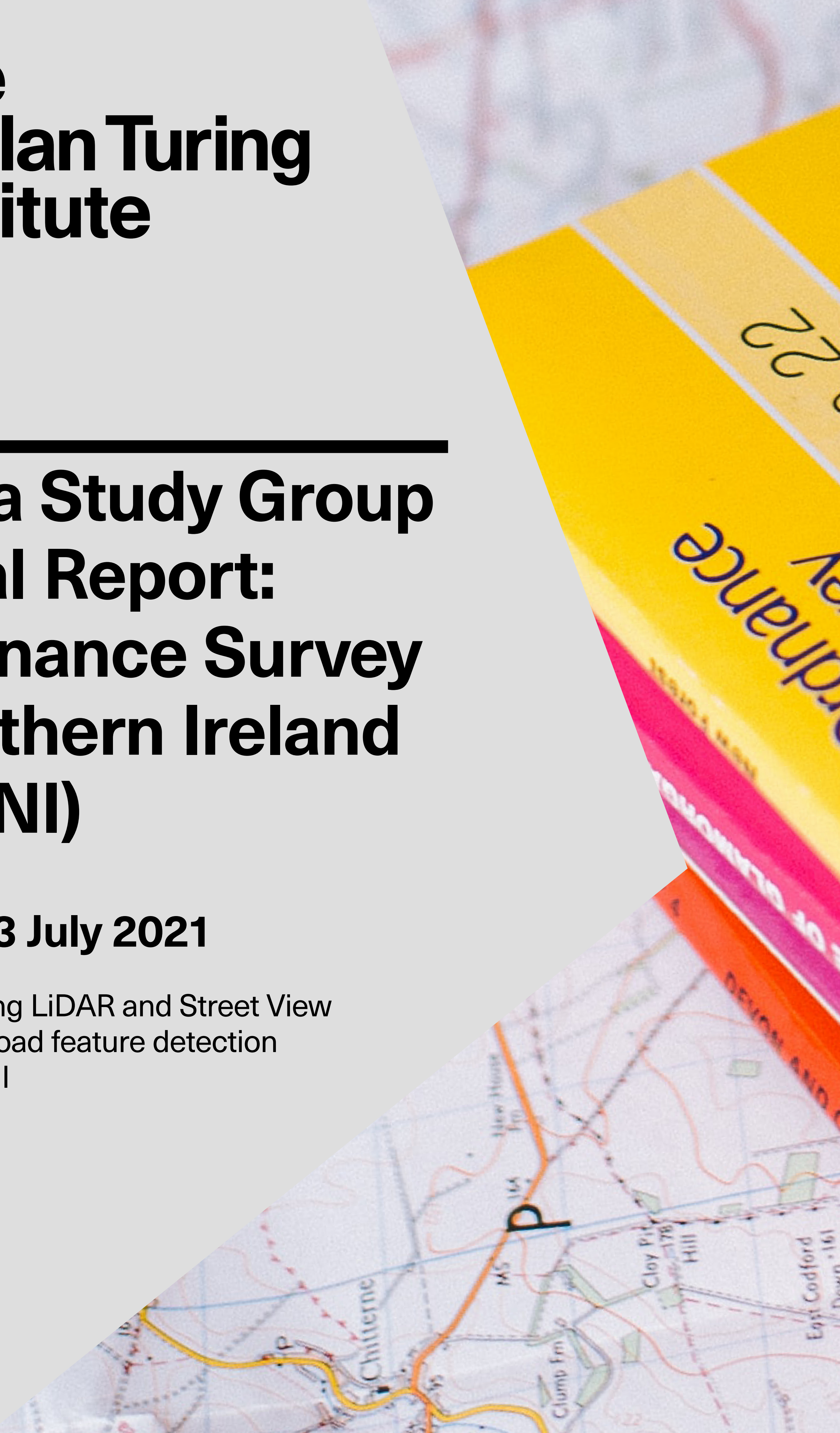
The Alan Turing Institute

Data Study Group Final Report: Ordnance Survey Northern Ireland (OSNI)

12 – 23 July 2021

Leveraging LiDAR and Street View data for road feature detection with OSNI



https://doi.org/10.5281/zenodo.6498764

Contents

Glossary

EPSG EPSG Geodetic Parameter Dataset is a public registry for spatial measurement standards. See Irish Grid and Table 4 https://epsg.io/29903. 2, 18

GeoJSON Geographic extensions to the JSON data format. See https://geojson.org. 6, 15, 22, 59

Imagelet Tiny perspective views of a scene from different viewpoints. See [64] and Figure 23. 40–42

ImageNet Image database organised according to the WordNet hierarchy (currently only the nouns), in which each node of the hierarchy is depicted by thousands of images. See [30] and https://www.image-net.org. 41

Irish Grid The Irish Grid Reference System (EPSG:29903 - TM75) is a coordinate reference system for Ireland which OSNI use for the data provided for this DSG. See https://epsg.io/29903 and https://www.ordnancesurvey.co.uk/documents/product-support/irish-grid.pdf. 2, 18

Keras ML framework built on TensorFlow 2. See [9] and https://keras.io/. 46, 61

KMeans Clustering algorithm for k groups closest by a distance measure. See [51]. 54, 55, 63

LabelImg Graphical image annotation tool. See [56], Table 2 and https://github.com/tzutalin/labelImg. 18, 23

laspy Python library for managing LiDAR LAS and LAZ files. See [6] and https://laspy.readthedocs.io. 32

LASTools Toolkit for processing LAS datafiles. See [20] and https://rapidlasso.com/LAStools. 31, 33

LATTE Web tool for annotating LiDAR Point Cloud data. See [59] and https://github.com/bernwang/latte. 28, 34, 36–38

LAZ Lossless compression format for LAS files. See LAS and [28]. 2, 35

Leica Pegasus Vehicle mounted mapping system comprising a LiDAR sensor, a 24 MP (2×12 MP) $360°$ camera, up to 6 additional 12 MP CMOS cameras and various other sensors (like GPS coordinates). See https://leica-geosystems.com/en-gb/products/mobile-mapping-systems/capture-platforms/leica-pegasus_two-ultimate. 4, 9, 12, 16, 18, 23

Matplotlib Python library for creating static, animated, and interactive visualisations. See [27] and https://matplotlib.org. 9, 30, 32

MeanShift Algorithm to locate maxima of a density function. See [51]. 54–56, 63

Octree Tree-like structure of exactly eight children for each internal node. See http://www.open3d.org/docs/latest/tutorial/geometry/octree.html. 33

Open3D Library for rendering 2D and 3D vector graphics. See [68] and http://www.open3d.org/. 33, 34

OpenDataNI Portal for Northern Ireland public sector data, which is part of the Digital Northern Ireland initiative. See https://www.opendatani.gov.uk. 15, 22, 59

OpenGL Cross-language, cross-platform API for rendering 2D and 3D vector graphics. See https://www.opengl.org/. 9, 30, 33, 34

Oxford RobotCar Dataset Over 100 repetitions of a consistent route through Oxford, UK, captured over more than a year. See [37, 3], Figure 12 and https://robotcar-dataset.robots.ox.ac.uk. 27, 29

Plas.io Web interface for visualising geospatial data. See https://plas.io/. 34–36

Point Cloud Spatial data points commonly used for LiDAR Point Clouds in three dimensions (X, Y, Z). 2–4, 6, 7, 9, 10, 12, 14, 18, 19, 23, 25, 28, 30–36, 38, 51–54, 57, 58, 60, 61, 63, 66

PointCNN Framework for feature learning from point clouds. See [32]. 14, 63, 64

PointNet Point Cloud data Voxel architecture for object classification, part segmentation and scene semantic parsing. See [8] and https://github.com/charlesq34/pointnet. 10, 14, 15, 60–63, 65

QGIS GIS (formerly Quantum GIS) application for viewing, editing, and analysing geospatial data. See [45]. 9, 30, 31, 33, 52

Rand Index Similarity measurement between two data clusters. See [53]. 58

Sheet 75 Rural region of Northern Ireland along the Glenelly Valley, running from Plumbridge to Sperrin. See Track and Figure 6. 4, 19, 21–23

Skymask Polar plot of a structures' silhouette following Skyplot [38]. See [62]. 27, 28

Skyplot Illustration of GPS satellite trajectories over a given ground site. See [38]. 3

SphereNet Framework for applying ML to spherical images. See [11]. 14

Streetscape.gl Toolkit to visualise autonomous vehicle XVIZ protocol data. See [57] and https://github.com/aurora-opensource/streetscape.gl. 34, 38, 39

TensorFlow ML and AI platform. See [12] and https://www.tensorflow.org. 2, 14, 46, 61

Track Leica Pegasus data provided by OSNI for this DSG, denoted by B, C, D, E, F, H, I, L, N, P and Q. See Sheet 75 (rural), NW200 (urban) and Figures 4, 5 and 6. 3, 7, 20–23, 27, 31, 32, 36, 45, 46, 49, 52, 57, 58

UrbanLoco Full sensor suite dataset for mapping and localisation. See [62] and https://github.com/weisongwen/UrbanLoco. 27, 28

VGG16 CNN Architecture for real time detection of objects in images developed by VGG. 16 refers to the number of layers. See [52]. 13, 14, 41, 65

Voxel Single sample or data point in a three-dimensional grid. 3, 51, 52

VoxelNet 3D Point Cloud feature extraction and bounding box prediction. See [69]. 14

Waymo Open Dataset Datasets originally constructed for developing self-driving cars. See [54] and https://github.com/waymo-research/waymo-open-dataset. 16, 17, 27

WebGL JavaScript API for rendering interactive 2D and 3D graphics within a web browser. See https://www.khronos.org/webgl/. 9, 30, 34, 37, 38

Whitebox Geospatial analysis and data visualisation tool. See [35] and https://www.whiteboxgeo.com/. 31

WordNet Lexical database of English. Nouns, verbs, adjectives and adverbs are grouped into sets of cognitive synonyms, each expressing a distinct concept. See https://wordnet.princeton.edu/. 2

XVIZ A protocol for real-time transfer and visualization of autonomy data. See https://avs.auto/#/xviz/. 4, 38

Acronyms

2D Two-dimensional. 9, 12, 15, 17–19, 22–27, 30, 31, 33, 40, 65

3D Three-dimensional. 9, 10, 12, 14–19, 27–29, 31, 33–38, 51, 58, 60, 63, 66, 67

AI Artificial Intelligence is machine approximation of human or animal intellect. 4, 7, 67, 68

API Application Programmable Interface. 3, 4

ASPRS The American Society of Photogrammetry and Remote Sensing maintains the LAS format OSNI provided for this DSG and is a member of the International Society for Photogrammetry and Remote Sensing (https://www.isprs.org/). See https://www.asprs.org. 6, 16, 19

cm centimetre. 18

CMOS Complementary Metal–Oxide–Semiconductor active pixel sensor. 2

CNN Convolutional Neural Network is a class of artificial Neural Network models commonly used in computer vision tasks. See R-CNN. 4, 7, 13, 14

COCO Microsoft Common Objects in Context (also MS-COCO) is a large-scale object detection, segmentation, and captioning dataset [34, 33]. See https://cocodataset.org. 44, 46, 47

DBSCAN Density-Based Spatial Clustering of Applications with Noise is a data clustering algorithm to group points that are closely packed together (points with many nearby neighbours). See [49]. 55–58, 63

DEM Digital Elevation Model is a representation of the bare ground (bare earth) topographic surface of the Earth excluding trees, buildings, and any other surface objects. 15

DSG Data Study Group is a 'collaborative hackathon' on research topics. See https://www.turing.ac.uk/collaborate-turing/data-study-groups. 2, 4, 5, 11, 12, 16, 17, 29, 33, 34, 65

DSM Digital Surface Model is a collection of points of bare earth including features such as buildings and vegetation. 15

DTM Digital Terrain Model is a grid of regularly spaced points of earth surface height excluding elements like buildings and plants. 15

GB Gigabyte means 1000 MB of data. 18, 19

GIS Geographic Information System tools process, analyse and present geographic information. See https://www.ordnancesurvey.co.uk/business-government/tools-support/gis/what-is-gis. 3, 27, 30, 31

GNSS Global Navigation Satellite System. See http://www.unoosa.org/oosa/en/ourwork/icg/icg.html . 28

GPS Global Positioning System. 2, 3, 16, 18, 19, 27

GPU Graphics Processing Unit is very efficient hardware designed to manipulate computer graphics and process images. 29, 45, 47

GUI Graphical User Interface is a means for users to interact with electronic devices through graphical icons. 9, 30, 35, 38

hr hour. 16, 18, 68

IMU Inertial Measurement Unit comprises a body's specific force, angular rate, and sometimes the orientation of the body via measurement devices including accelerometers and gyroscopes. 16

IoU Intersection over Union is a measure of the accuracy of an object detector by finding the overlap regions between the predicted and ground truth masks divided by the union of the two. 7, 45, 48, 49

JADE The Joint Academic Data Science Endeavour is a Tier 2 facility supporting research in ML. See https://www.jade.ac.uk/. 29, 45

JSON JavaScript Object Notation is a lightweight, human-readable data interchange format. See GeoJSON and https://www.json.org. 2, 46, 47

KAIST The Korea Advanced Institute of Science and Technology is a national research university located in Daedeok Innopolis, Daejeon, South Korea. See https://www.kaist.ac.kr/. 27

KITTI The Vision Benchmark collaboration between the Karlsruhe Institute of Technology and the Toyota Technological Institute of Chicago [16, 18] (KITTI) is a suite of computer vision benchmark datasets collected by a camera mounted vehicle http://www.cvlibs.net/datasets/kitti/. 16, 17, 38, 44

km kilometre. 16–18, 27

LAS LASer format is a LiDAR Point Cloud data file format maintained by ASPRS. See https://www.asprs.org/divisions-committees/lidar-division/laser-las-file-format-exchange-activities. 2, 5, 6, 19, 30, 32, 35

LiDAR Light Detection and Ranging (or Laser Imaging, Detection, and Ranging) is a means of detecting the distance to a surface by timing the reflection of a laser back to the point from which it was emitted. See https://oceanservice.noaa.gov/facts/lidar.html. 2, 3, 6, 7, 9, 10, 12–19, 22, 25–31, 33–38, 51, 54–56, 59, 60, 63–66

LPS Land & Property Services manages land and property in Northern Ireland and includes OSNI. See https://www.finance-ni.gov.uk/land-property-services-lps. 12

m metre. 16, 18

mAP Mean Average Precision is a measure of prediction accuracy where higher is better. See [14] for an example of the metric applied in object detection and IoU. 46, 49

MB Megabyte is 1 million bytes of data. 5, 30

ML Machine Learning algorithms make predictions or decisions without being explicitly programmed to do so. See AI. 2–4, 6, 9, 12, 14, 15, 29, 33, 63, 66–68

MP Megapixel is one million pixels (points with a numeric colour value). `MP` is often used to indicate the number of pixels in an image from a camera or sensor. 2, 7, 16, 18

NEON National (U.S.) Ecological Observatory Network—part of the U.S. National Science Foundation—is an observation facility providing data on U.S. ecosystems. See [40] and https://www.neonscience.org/. 33–36

NW200 Northwest 200 is an urban section of Northern Ireland roads connecting Portrush, Portstewart and Coleraine. See Track and Figure 5. 4, 19, 20, 22

OSNI Ordnance Survey of Northern Ireland. 2, 4–6, 8–13, 15–18, 22, 23, 25–28, 30, 31, 33–36, 38, 43, 44, 59, 65, 66, 68

pptk Point Processing Toolkit is a Python library for visualising and processing 2D and 3D Point Clouds. See [36]. 33, 34

PyPI The Python Package Index provides publicly available Python software. See https://pypi.org. 32–34

R-CNN Region Based Convolutional Neural Network are a family of object detection models which employ a “recognition using regions” paradigm. See [19] and CNN. 5, 14, 45–49, 65

RANSAC Random Sample Consensus is an iterative algorithm for robust fitting of models of data containing outliers. It can be used to detect outliers in a dataset. See [15]. 54, 58

RGB Red Green Blue is an active colour model commonly used in digital images. 41, 64

RPN Region Proposal Network is a fully convolutional network that simultaneously predicts object bounds and objectness scores at each position. 48

SaNE Smart Annotation and Evaluation Tools for Point Cloud Data is a `LiDAR` annotation tool. See [1] and https://github.com/hasanari/sane. 38

VGG Visual Geometry Group is a computer vision research group. See https://www.robots.ox.ac.uk/~vgg/. 4, 8, 47

VIA VGG Image Annotator is a manual web interface annotation package for image, audio and video. See [13] and `https://www.robots.ox.ac.uk/~vgg/software/via`. 47

VM Virtual Machine means virtual computing system, where all the aspects of a computer—including hardware—are run virtually as software. OSNI data was securely analysed for this project within VMs managed by the University of Leeds. 8, 29, 30, 33–35

YOLO You Only Look Once is a real time object detection algorithm. See [47]. 13, 14, 45, 46, 65

Executive Summary

Challenge Overview The Ordnance Survey of Northern Ireland (OSNI) mission is to provide high quality geospatial data. Historically this has been for 2D mapping, but modern survey techniques and increasing user requirements have shifted focus toward 3D data. Since 2019, OSNI has operated a vehicle mounted Mobile Mapping System (Leica Pegasus:Two Ultimate Mobile Mapping System) across Northern Ireland capturing 3D Point Cloud data and spherical street view imagery.

The range of potential applications is significant, including urban planning, asset identification and management, automating identification of road sign changes for navigation and transport network datasets, identifying feature locations such as scenic views, drainage, potholes and road surface quality, street furniture maintenance, 5G network planning and managing autonomous vehicles. While availability and accessibility of this kind of raw data is improving, there are significant technical challenges in deriving insights from the richness of this dataset.

To address these challenges this project seeks to explore the potential of OSNI's highly detailed Light Detection and Ranging (LiDAR) and imagery data via Machine Learning (ML) and data science methods, with a focus on developing pipelines to visualise, classify and identify road features like drainage which could potentially help various government authorities better monitor road infrastructure. There are many other potential applications for the sort of data OSNI collects, and we hope some of the pipelines and visualisations explored below can aid broader applicability. Below are the results from each of the streams of work conducted.

LiDAR Data Visualisation An important element of working with LiDAR data is visualisation. To that end some non-commercial tools that can be used to aid understanding and presenting the data were explored. This included Graphical User Interface (GUI) tools such as QGIS (Section 3.1), standard Python visualisation libraries such as Matplotlib (Section 3.2) and the libraries associated with OpenGL (Section 3.3) and WebGL (Section 3.4). We detail the installation, advantages and disadvantages of each approach for use in applications OSNI might provide and maintain.

Spherical Image Pipeline In this section several approaches to detect drainages using 2D spherical images were explored, including

i transfer learning to classify whether a part of the image has drainage or not (Section 4.1)

ii exploring a pre-trained road segmentation model in masking the road sections of street images (Section 4.2)

iii training both a standard object detection model (Section 4.3), as well as an object instance segmentation model to identify and localise drainage from street images (Section 4.4).

Despite the size of road drainage and limited ground truth labels, the results of the different approaches were generally promising.

There are areas to improve for all four approaches, such as leveraging data continuity for object tracking and the need for more ground truth labels to help improve accuracy and model generalisation. A strength of this approach is it only requires image files (smaller file sizes relative to LiDAR data) which has a wealth of established open source methods and pipelines to process the data (elements of which we have refined, including a means to project the spherical images to regular images). These methods can also be easily adapted to retrieve geographical information at the street-level such as pot holes, trees and building facades in the future.

LiDAR Pipeline In this section, we explored the 3D Point Cloud dataset through three standard tasks:

i testing different approaches to Point Cloud sampling to reduce computation time (Section 5.2).

ii exploring different unsupervised approaches to cluster Point Cloud data (Section 5.3).

iii examining a standard Point Cloud classifier (PointNet) to determine whether a Point Cloud sample has a drainage or not (Section 5.4).

Given the data constraints (e.g. size and complexity), our initial results from training the 3D Point Cloud classifier were encouraging. However, there are prospects for improvement, particularly on the exploration and visualisation of Point Cloud clusters and the lack of open and accurate ground truth labels. The strength of analysing this data is its richness in potentially collecting novel 3D information from our environment.

Limitations A number of limitations constrained our study. The most prominent across both streams of work is the lack of labelled data. Increasing the amount of annotated data should improve the model accuracy and generalisation. Second is temporal: we did not have enough time to take advantage of the richness, continuity and multi-modality of the dataset, and there is good reason to believe that by fusing the Point Cloud and image data a more accurate model could be developed.

Recommendations There is great potential for OSNI to use this data to aid government departments in planning policies with wide application for the general public. Developing a pipeline to measure a wide range of geospatial features is essential to that mission.

We have a number of recommendations to achieve this. The first is to further explore linkages (eg. using meta-data) between the image and LiDAR datasets. This will aid the development of a process to fuse the two models. The second is to increase the quantity and diversity of the training data by using advanced visualisation and annotation tools. The third is to exploit the continuity of the data for object tracking and the vast unlabelled imagery for semi-supervised and active learning. The fourth and final is to release data and methods processed and developed within this Data Study Group (DSG) for the mutual benefit of the growing research in this domain. For example: pre-trained models can be used to retrieve useful information with minimal supervision, and the more researchers apply and improve these methods the more they could in turn aid the community and OSNI. All of these directions are possible avenues for future, publishable research.

1 Introduction

1.1 Ordnance Survey Northern Ireland

Ordnance Survey of Northern Ireland (OSNI) is the official mapping agency for Northern Ireland which produces highly accurate geographic mapping data. OSNI—a part of Land & Property Services (LPS) within the Department of Finance of Northern Ireland—has historically focused on 2D mapping but has recently shifted to more modern survey techniques and increasing user requirements for 3D data [43]. OSNI also aim to offer a Geospatial Survey Service to facilitate road health, such as identifying feature locations for drainage, potholes and road surface quality for the Department for Infrastructure.

1.2 Project Objectives

This project seeks to explore the potential of this highly detailed Light Detection and Ranging (LiDAR) and imagery data via Machine Learning (ML) and data science methods, with a focus on developing pipelines to classify and identify urban features like drainage within street images and Point Cloud data. This project specifically focused on the detection of street drainages which can be used by the public authority such as the Department of Transport to better monitor its road infrastructure. In addition, the method and its pipeline can be adapted to detect other urban features which would be a valuable asset for a range of government departments. We hope some portions of the project can be released publicly for general use. More specific questions that we hope to address with this analysis include:

- Explore visualisation and annotation methods of LiDAR data.
- Evaluate the effectiveness of applying standard ML models on street images (including spherical) and Point Cloud (LiDAR) data for the classification or localisation of small road objects such as drainage.
- Explore multi-modal methods for detection of small objects like drains.
- Do spherical images perform better or worse than normal view imagery for object detection?

1.3 Literature Review

To determine the best approach for this DSG challenge we conducted a concise literature review. This familiarised us with the Leica Pegasus (via official manuals), street imagery and LiDAR technology, as well as data science methods that are being applied on images (including spherical ones) and LiDAR data. We also found promising work on fusing

LiDAR and Spherical Image models that could potentially combine the datasets provided by OSNI for future development.

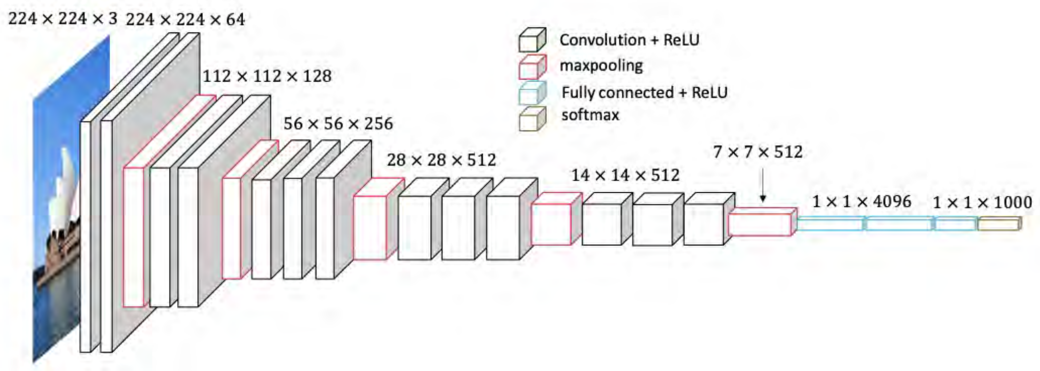


Figure 1: VGG16 Architecture [52]. Figure from [60].

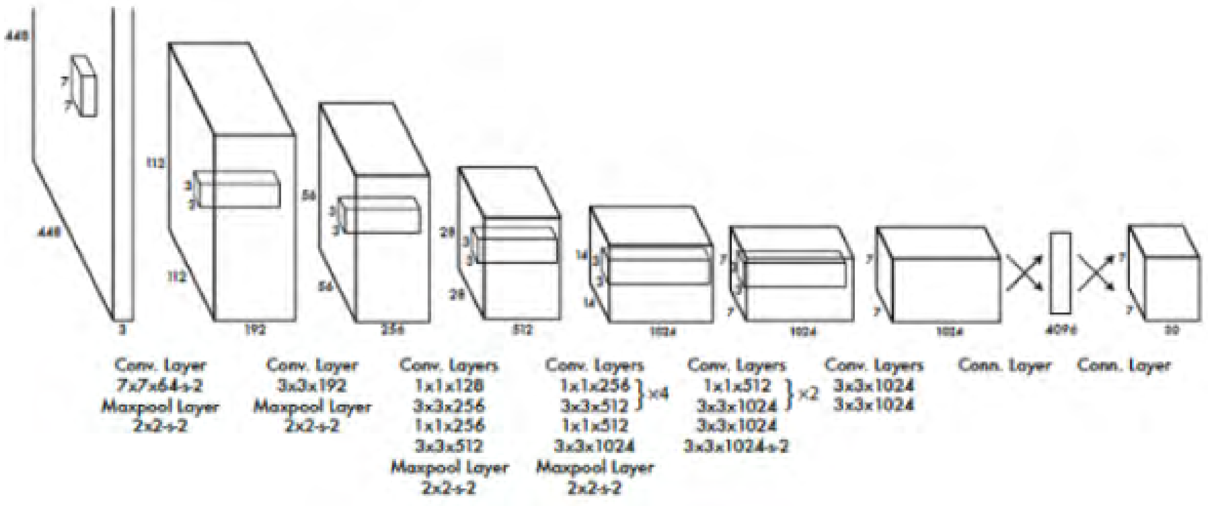


Figure 2: You Only Look Once (YOLO) Architecture and Figure from [47].

Standard images models Convolutional Neural Network (CNN) [30] is a type of Neural Network model that is commonly used for computer vision tasks such as image classification (to classify whether the image is a particular class or not), object detection (to localise an object in an image), and semantic segmentation (to classify each pixels of an image). To begin, we looked at standard computer vision models for these tasks including VGG16 [52] for image classification (Figure 1), YOLO [47], YOLOX[1] [4, 17] for object detection and Masked Region Based Convolutional Neural Network (R-CNN)[2] [24] for object instance segmentation (Figure 3).

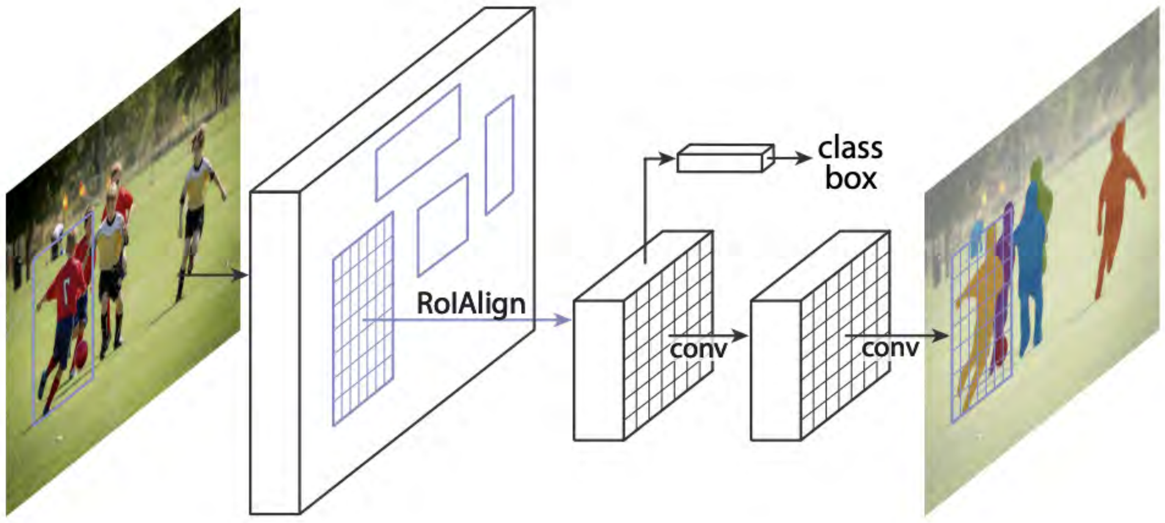


Figure 3: MaskR-CNN example from [24]

Spherical Images models Standard CNN models are not necessarily well suited for spherical images as the natural projection surface is a sphere which cannot be unwrapped to a plane without introducing significant distortions (particularly in polar regions). Various researchers tried to tackle this. For example, SphereNet [11] applied a deep learning framework which encodes invariance against such distortions explicitly into a convolutional neural network.

Standard `LiDAR` models There are several ML models in the literature that deal with Point Cloud data, for example PointNet [8], VoxelNet [69] and PointCNN [32]. Due to time constraints we focused on PointNet, a deep learning architecture illustrated in Figure 4, for Point Cloud classification and segmentation. Please see [8] for further details. An implementation of PointNet using TensorFlow 2.0 can be found in [21].

For a more comprehensive review regarding `3D LiDAR` data, please see [22] which highlights recent ML models and architecture as well as fundamental challenges that arise from analysing this type of data.

[1]See https://github.com/Megvii-BaseDetection/YOLOX
[2]See https://github.com/facebookresearch/Detectron

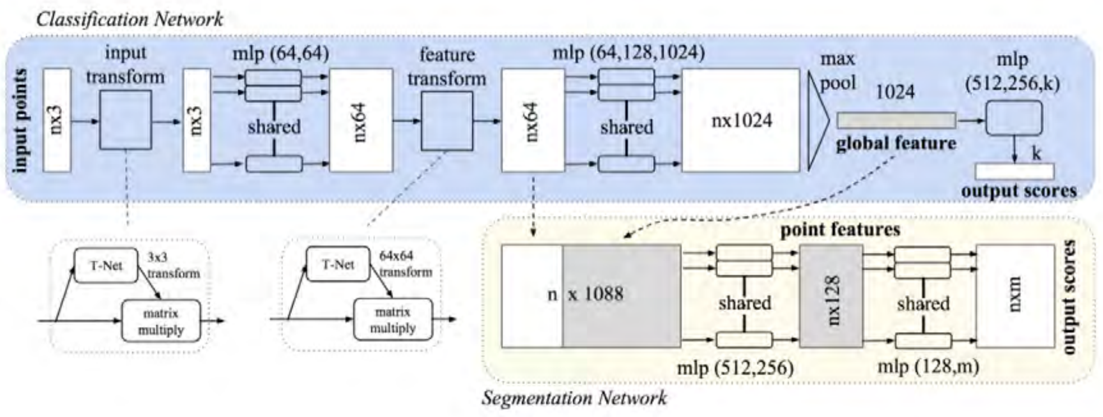


Figure 4: PointNet Architecture from [8]

Fusion of LiDAR and Spherical image data There has been growing interest in examining the fusion of LiDAR and spherical image data for different ML tasks [2, 7, 23, 65]. Due to the time constraints of this DSG, we did not examine this area in great detail. However, there are opportunities here as the data captured from the LiDAR equipment have been synchronised. Further research is necessary to examine how the two types of data can be fused and the architecture that can take advantage of this fusion. In our study, we have explored projecting the 2D bounding boxes into 3D space as an example. Please see Section 2.3 for more details.

Open access to other Geospatial and LiDAR data There are several open access Geospatial and LiDAR datasets and in fact during the design and development phases of the LiDAR pipeline, in addition to the data provided by OSNI, we briefly explored using open access drainage data (GeoJSON) from OpenDataNI Portal[3]. Regarding the open access to other typologies of LiDAR data (airborne, satellite) as well as other geospatial products, currently there are some consolidated programmes in the U.K., EU and in the U.S. maintained by the Environment Agency[4], the European Environment Agency[5] and United States Geological Survey (USGS)[6] respectively. It is possible to download geospatial products such as a Digital Elevation Model (DEM), Digital Terrain Model (DTM), Digital Surface Model (DSM) data that could facilitate the monitoring of large urban and rural areas and integration with ground-based and mobile mapping systems.

[3]OpenDataNI: https://www.opendatani.gov.uk/dataset/drainage-asset

[4]EA: https://data.gov.uk/dataset/f0db0249-f17b-4036-9e65-309148c97ce4/national-lidar-programme

[5]EEA: https://land.copernicus.eu/imagery-in-situ/eu-dem/eu-dem-v1.1

[6]USGS: https://prd-tnm.s3.amazonaws.com/LidarExplorer/index.html/

	Waymo Open Dataset [54] [7]	KITTI [18]	OSNI Data DSG
Tools	• Camera • Waymo Driver sensors	• 2 Colour Video Cameras • 2 Greyscale Video Cameras • GPS/IMU navigation system • 3D Velodyne laser scanner	Leica Pegasus: • LiDAR sensor • 1 spherical 24 MP camera • 6 mounted 12 MP cameras
Object Data	• Vehicles • Cyclists • Person • Pedestrians • Residential	• Road • Campus • City	• Drainage • Road Signs
Recording Locations	• San Francisco • Mountain View (US) • Phoenix	• Karlsruhe (Germany)	• Northern Ireland
Location Diversity	• Downtown • Daylight • Pedestrians • Diverse Weather • Suburban • Night • Cyclists • Constructions	• Urban • Highway • Rural • Daylight	• Urban • Rural • Daylight
Labelled Data	• Vehicles • Cyclists • Pedestrians • Signs	• Car • Truck • Person (seated) • Tram • Van • Pedestrian • Cyclist • Misc	• Drainage • Road Signs • ASPRS LiDAR Classification
Map Visited Areas	• Phoenix 40 km^2 • San Francisco & Mountain View 36 km^2 • Total 76 km^2	• 6 hrs of traffic[8]	• 1800 km
Avg. Dist. Between Images	No rate found during the DSG.	• 20 m	• 3 m

Table 1: Comparison of OSNI LiDAR DSG challenge dataset with Waymo Open Dataset [54] and KITTI dataset [18].

Related works that provide open datasets for 2D images, 3D object detection and 3D tracking include the Waymo Open Dataset[9] [54] and the KITTI[10]. The Waymo Open Dataset has approximately 12 million 2D bounding box labels (with tracking IDs on camera data) and 12.6 million 3D bounding box labels (with tracking IDs on LiDAR data) [54]. The main website associated with the KITTI provides various types of dataset from raw data [18], to processed stereo data, visual odometry, and a road dataset that demarcate lanes within street scenes [16]. Table 1 compares the different types of tools used: object data, diversity type of location, where the recording located, number of labelled data, kilometre (km) of visited area in the Waymo Open Dataset [54], KITTI [18] and the OSNI data used in this challenge.

During the DSG we considered ways of using the standard architectures cited above for both image and LiDAR data. For the future we recommend exploring the use of these assets for both improving the performance of classification models and on exploring the fusion of the two streams of data for multi-modal modelling.

[7]Formerly Google Autonomous Vehicle

[8]Specification not available in km.

[9]The Waymo Open Dataset is available at https://waymo.com/open/

[10]KITTI is available at http://www.cvlibs.net/datasets/kitti/

2 Data Overview

2.1 Dataset Description

Since 2019, OSNI has operated a vehicle mounted Leica Pegasus across 1800 km of Northern Ireland roads. This multi-sensor recording platform is equipped with a GPS satellite dish, four 12 MP cameras placed along the perimeter of the sensor, one $360°$ spherical camera at the top and a LiDAR laser scanner. The LiDAR laser sensor determines the distance to the first object on its path by emitting a laser pulse into the surrounding environment and measuring the time for the reflected light to return to the receiver.

Repeating this process millions of times per second in a $360°$ range creates a precise, 3D visualisation of the surrounding area known as a Point Cloud, with approximately 5 cm point spacing (assuming driving at 40 km/hr). At the same time, the four cameras capture images in a manner synchronous with the LiDAR sensor, meaning that they take images when the LiDAR scanner is at the centre of their field of views. As such, the multi-modal sensor captures consistent 3D Point Cloud data, 2D spherical street view imagery and normal view imagery of roads and their surroundings, all of which are geo-referenced to that framework allowing one to know the exact location of each feature. The collected information is aimed to be integrated together through multi-modal techniques to deliver high accuracy of measurements. Please see Table 1 for comparison with other LiDAR datasets.

Image Classification		**Count**
Drainage[11]	Bounding Box	467
	No Drains	502
	Total	**969**
Road Signs[12]		227
Unclassified		20662
Total Images		**21631**

Table 2: Number of images in the OSNI dataset, split by classification type (contains drainage; does not contain drainage and unknown), and a dataset of road signs. Images were also provided with (X, Y) coordinates (Z not included) in the Irish Grid EPSG:29903 - TM75 coordinate system (see https://epsg.io/29903). Please see Table 4 for more detail descriptions of the training data.

OSNI provided two datasets. First: approximately 76 GB of colourised data of spherical images collected at 3 m intervals in JPEG format listed in Table 2. The 2D imagery is

[11]OSNI staff used LabelImg to draw bounding boxes around drains.

[12]These were not used in the analysis in this report.

accompanied with additional location files, specifying the (X, Y) location of each image which can aid integrating the two data-sets. The second is approximately 91 GB of colourised, 3D Point Cloud data. This data is provided in a format with columns listed in Table 3.

Attribute	Description
X	Coordinate of observation $X = (X_{\text{record}} * X_{\text{scale}}) + X_{\text{offset}}$
Y	Coordinate of observation $Y = (Y_{\text{record}} * Y_{\text{scale}}) + Y_{\text{offset}}$
Z	Coordinate of observation $Z = (Z_{\text{record}} * Z_{\text{scale}}) + Z_{\text{offset}}$
GPS	GPS time point of observation
Intensity	Amount of light energy recorded by the sensor
Blue	Blue image channel value
Green	Green image channel value
Red	Red image channel value
Return Number	Which sequential pulse received
Number of Returns	Total number returned for an emitted pulse[13]
Scan Direction Flag	Direction of the scanner mirror at the time of the output pulse
Edge of Flight line	Last point on a given scan line before change of direction
Scan Angle Rank	Scanner angle at time of output pulse (in $[-90°, +90°]$)
Classification	Type of point[14] such as `Ground`, `Building` and `Water`
Point Source ID	The file from which the point originated
User Data	Optional additional data (not used)

Table 3: Columns in the LAS data Version 1.2 format [31] managed by the ASPRS.

The data provided has been collected while driving along as shown in Sheet 75 (rural) of Figure 6 and in sheet Northwest 200 (NW200) (urbanised) of Figure 5. Given the way the data were captured, they exhibit continuity in the sense that sequential images have significant overlap and exhibit similar weather and traffic patterns. Examples of labelled 2D imagery and LiDAR data is shown in Figure 7.

[13] LiDAR works like ultrasound. Every time there is a change in the density of the material the pulse is travelling through, part of it is transmitted and part of it is reflected back towards the sensor. The quantities 'return number' and 'total pulse returns' capture precisely this phenomenon.

[14] See page 10 of https://www.asprs.org/a/society/committees/standards/asprs_las_format_v12.pdf for details.

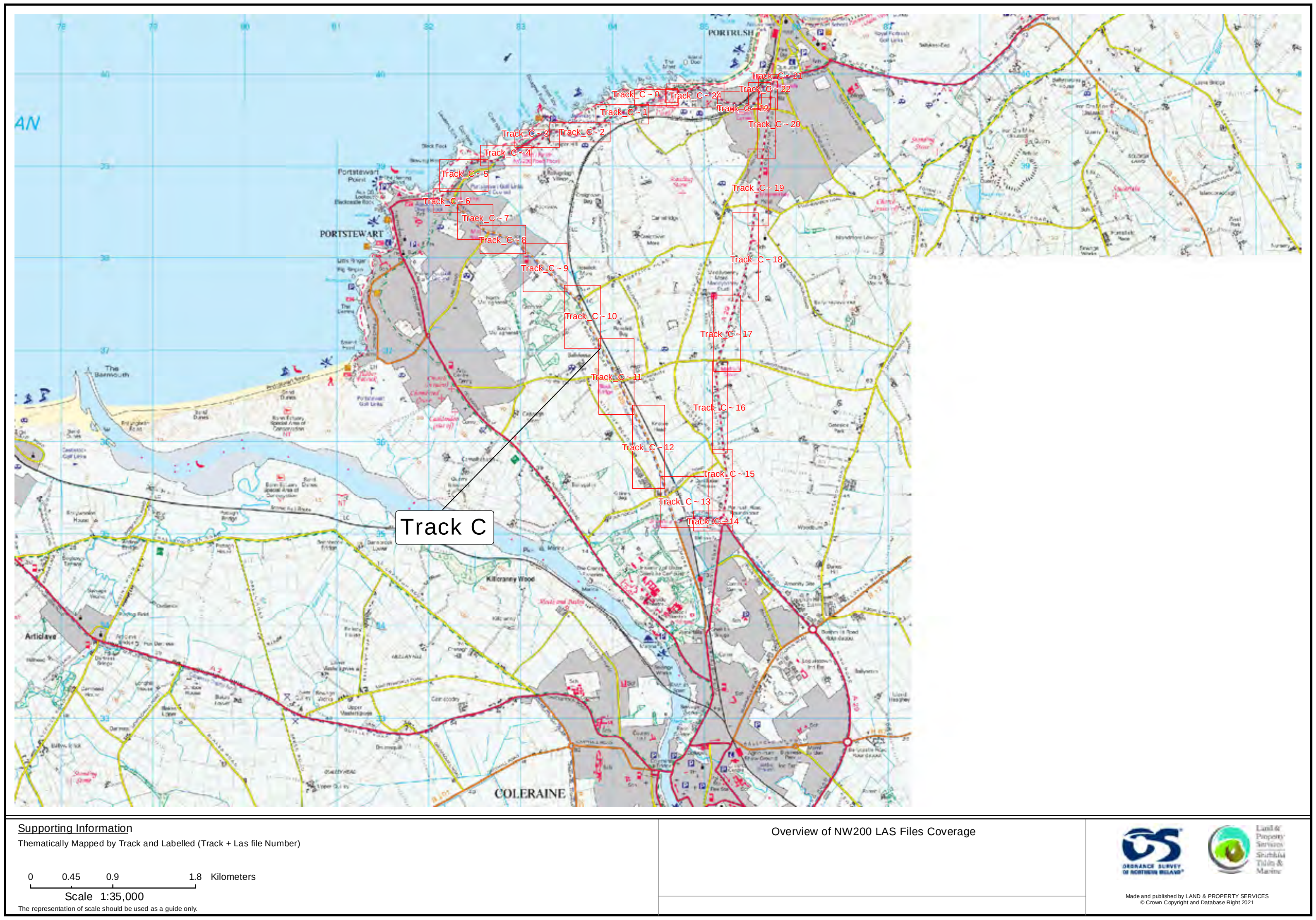


Figure 5: Visualisation of the path along which data has been collected in sheet NW200 (urban) with labelled Tracks.

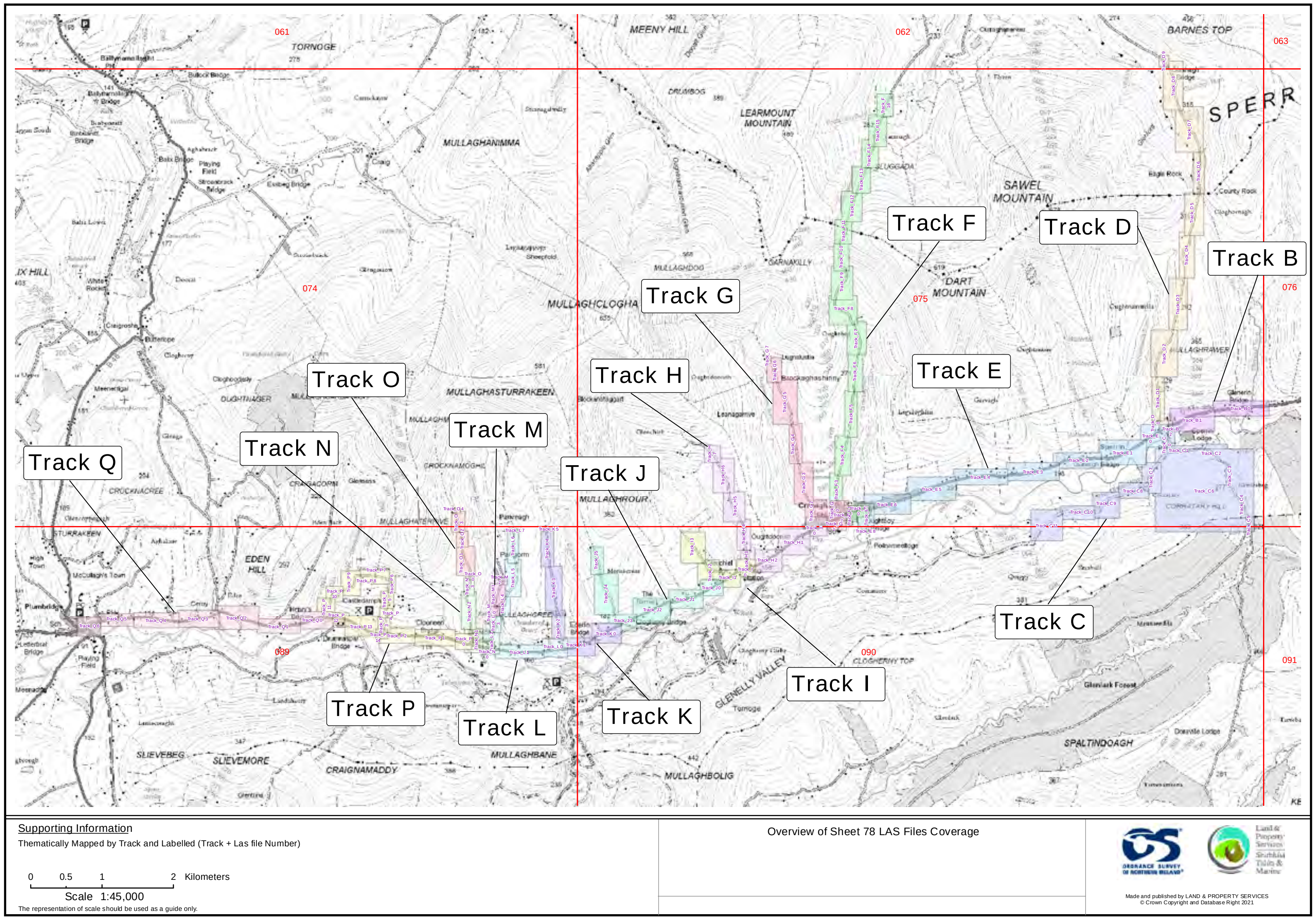

Track B
Track C
Track D
Track E
Track F
Track G
Track H
Track I
Track J
Track K
Track L
Track M
Track N
Track O
Track P
Track Q
Supporting Information
Thematically Mapped by Track and Labelled (Track + Las file Number)
0 0.5 1 2 Kilometers
Scale 1:45,000
The representation of scale should be used as a guide only.
Overview of Sheet 78 LAS Files Coverage
Made and published by LAND & PROPERTY SERVICES
© Crown Copyright and Database Right 2021

Figure 6: Visualisation of the path along which data has been collected in Sheet 75 (rural), with labelled Tracks (B–Q).

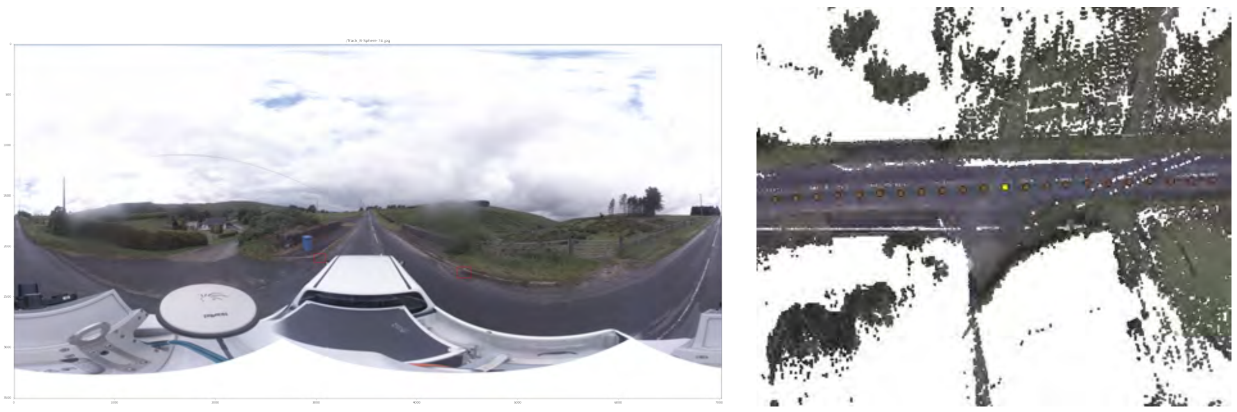


Figure 7: A sample of the data provided: a `2D` spherical image (left) and a segment of a `LiDAR` dataset (right). The red box corresponds to a bounding box provided by OSNI as part of the labelled data. The yellow dot corresponds to the location where the image on the left was taken. Note that labels were only provided for a limited set of `2D` imagery.

Training Test Split In order to achieve meaningful comparisons between the various models we performed a spatial train and test split using the Tracks with labelled images. The number of images in each Track can be seen in Table 4. The spatial location of the Tracks can be seen in Figure 6 and Figure 5. In particular, the labelled data in Tracks B, D, F, I, L, N, P from Sheet 75 and C from NW200 have been used for training purposes and the data in Tracks E, H, Q (Sheet 75) have been designated for testing, corresponding roughly to an 80-20 split.

In some experiments we sampled or used the data in slightly a different fashion for reasons such as the constraint of computing time (for example see Section 4.4) and the need to relabel the data (see Section 5.4). We chose this split, over for example a temporal split, because of its simplicity and to reduce the possibility of sequential data leading to data leakage from the training set into the test set. However, it should be noted that this assumes that the distribution of images in the train and test set are the same, which might not be completely accurate given variable weather conditions and urbanisation levels.[15]

It should also be noted that for specific models explored in this report, some of the `2D` imagery had to be re-labelled due to both issues with the annotation file format as well as the inaccuracy of the bounding boxes after projection to rectangular images. In Section 2.3 we explored the transfer of the bounding boxes from the `2D` labelled imagery to the `LiDAR` data. Other approaches for labelling the `LiDAR` data included using unsupervised models on the raw `LiDAR`, which turned out to be very difficult, and importing the open source drainage data (`GeoJSON`) from OpenDataNI, which turned out to be unreliable due to non-uniform misalignment with the OSNI data. At the end,

[15]There was a discussion of considering k-fold cross-validation due to the lack of data.

Dataset	Track	Drainage	No Drainage	Train/Test Spilt	Total Images
Northwest 200	C	274	151	Training	4805
Sheet 75	B	18	42	Training	518
	C	–	–	–	1630
	D	0	42	Training	1852
	E	12	0	Testing	1579
	F	39	8	Training	2268
	G	–	–	–	1017
	H	14	0	Testing	1043
	I	8	0	Training	488
	J	–	–	–	917
	K	–	–	–	870
	L	23	91	Training	990
	N	7	0	Training	173
	M	–	–	–	423
	O	–	–	–	332
	P	15	51	Training	1508
	Q	57	117	Testing	1218

Table 4: The size of labelled datasets provided by OSNI with drainage labelling via LabelImg [56]. Track letters refer to those indicated on in Figures 5 and 6. Tracks C, J, K, M and O from Figure 6 were not labelled (as indicated by the '–' character).

manual labelling was used to create a more reliable labelled dataset for the Point Cloud data.

2.2 Project Spherical Images to Regular Images

Despite the Leica Pegasus system being equipped with normal view cameras (non-spherical), OSNI only provided spherical images. The reasoning for this was that it was less time consuming to annotate spherical images. However, this introduced distortion to the shape of the drainage, which could potentially affect the performance of the models that we studied. An attempt to quantify this difference was carried out in Section 4.1.

In order to avoid this, we pre-processed the `2D` spherical imagery using the Equirectangular-toolbox [39].

As shown in Figure 8, essentially the package takes as input the spherical imagery (which one can think of as living on a 2-dimensional, unit-radius sphere) and a point P (which corresponds to the point of the sphere from which one is looking from) and performs a gnomonic projection on a plane tangential to P. An example of this operation is shown

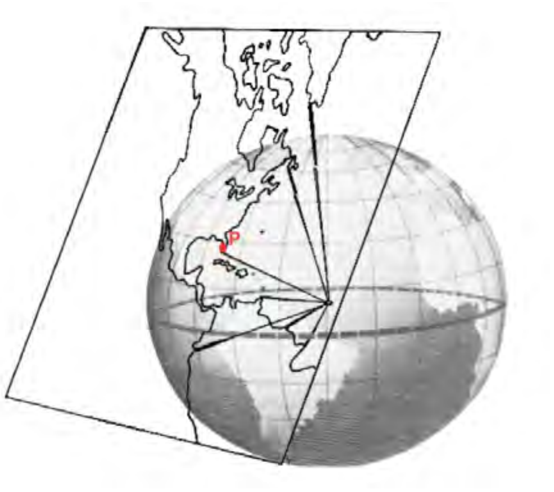


Figure 8: Gnomonic projection of 2D spherical image through point P Image from [5].

in Figure 9. The point $P = (\theta, \phi)$ takes values $\theta \in [0, 1]$, $\phi \in [0, 1]$ where $\theta = 0$ points upwards and $\theta = 1$ points downwards, while ϕ is the azimuthal angle corresponding to rotating around the car.

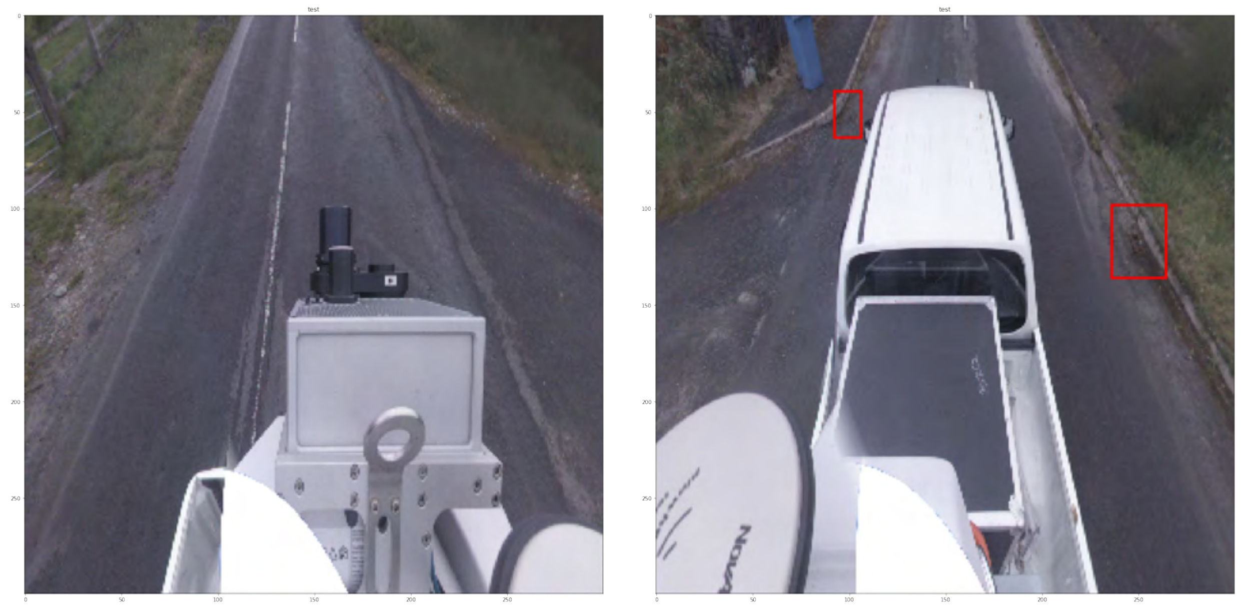


Figure 9: Example of gnomonic projection of spherical image in Figure 7 with $P = (0.75, 0)$ (left) and $P = (0.75, 0.5)$ (right).

Given that we are particularly interested in drains which are located close to the ground, we have used $\theta = 0.75$ for the majority of our data pre-processing. Furthermore, note that this process provides a natural way for data augmentation by projecting from different points around the car (using several values for ϕ). This process was used both for the spherical imagery and the bounding boxes provided. In the case of the bounding boxes, this introduced some inaccuracies in the sense that the boxes got slightly bigger than they should have been.

2.3 LiDAR and Imagery Merge

As mentioned above, the labelling provided by OSNI was only for the 2D spherical imagery. In particular, no labelling was provided for the LiDAR data. In this subsection we explore the possibility of transferring labels between imagery and LiDAR.

To begin with, we matched the 2D labelled imagery files with the LiDAR files that included the (X, Y) location[16] where the spherical image was taken—note that the altitude Z of the location where the image was taken was not provided and thus the Z coordinate was not used at all in this discussion.

Through a simple shift, we transferred the origin of the coordinate system to the location where the image was taken and we considered a unit-radius sphere centred at that location. Then, we converted the location of the bounding box on the spherical image to angles (θ, ϕ), focusing only on the azimuthal angle ϕ because of the reason mentioned above related to the Z direction. Having this at hand, we filtered all the Point Cloud data keeping only those that lie within the azimuthal angle range of the bounding box (plus a buffer). Given that the drains are not too far from where the image was taken, we also filtered on the radial distance, mainly to improve visualisation.

In Figure 10, on the top we show a (labelled) spherical image containing a drain and on the right we show the result of the procedure outlined above. In both images, ones can easily see the car, the pavement and the tree, while the drain is not particularly easy to identify.

The images above highlight issues regarding the detection of small size objects, and in particular drainage, through Point Cloud data. We see that, in certain cases, it can be particularly difficult to see the drains in the Point Cloud data as objects are both too small and can be occluded during the scan. This issue seems to be particularly pronounced in rural areas which requires further analysis to verify.

2.4 Limitations

As with all forms of LiDAR data, there are some general limitations in the OSNI dataset and analysis thereof, and in this section we discuss the limitations our team members have noted, and what can be done to alleviate them.

[16]Details of data included in LiDAR files—including coordinates—are listed in Table 3.

Variation of Urbanisation and Location The dataset collected from OSNI are mostly taken in both rural and low density urban locations. As such the model trained with OSNI data might not generalise to highly urbanised locations such as Hong Kong and San

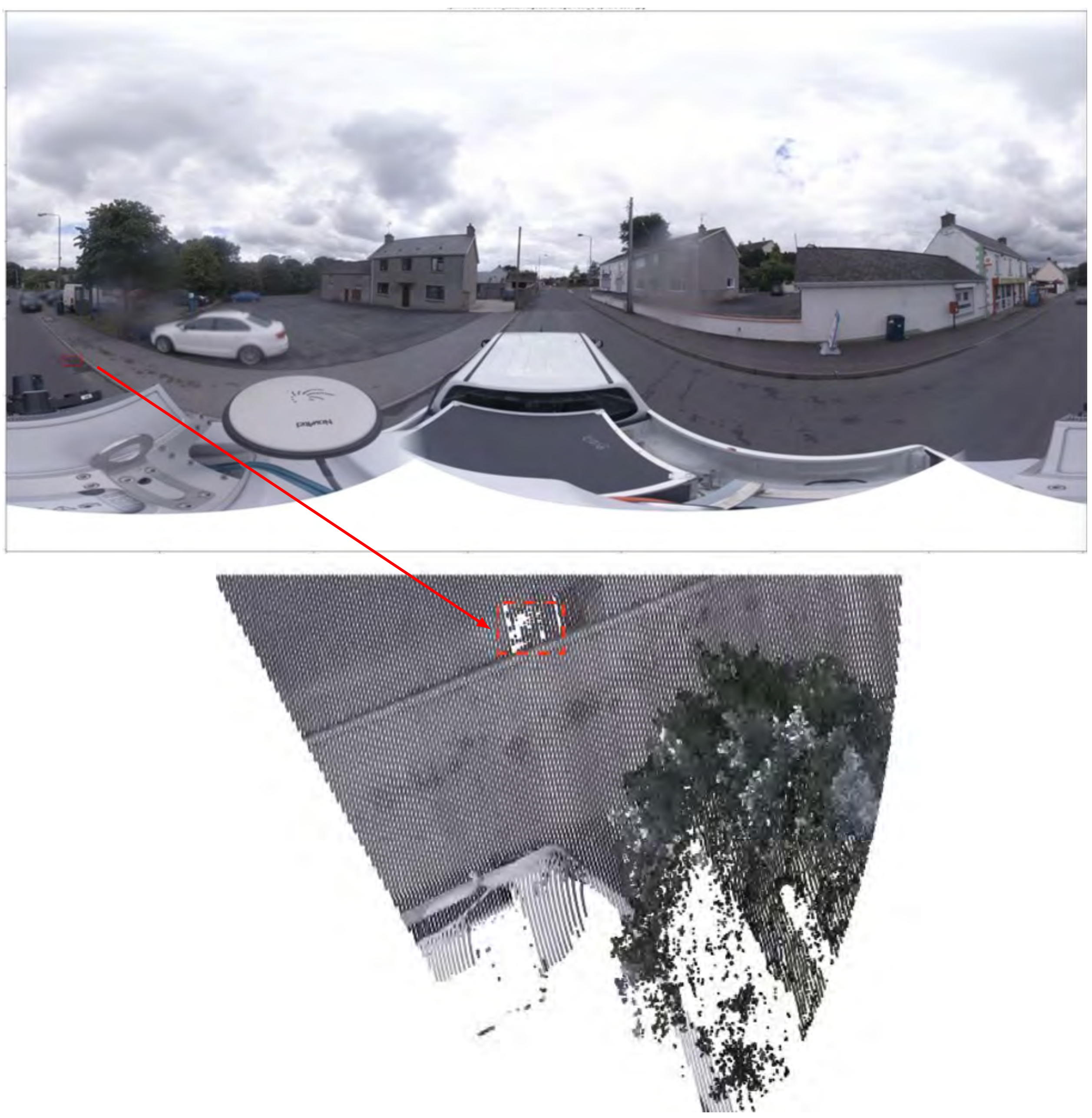


Figure 10: A labelled 2D spherical image including one bounding box (top) and a view of the LiDAR data corresponding to the region around that bounding box. The dashed box corresponds to where we expect to see the drain.

Francisco as captured in the UrbanLoco [62] and the Waymo Open Dataset [54].

Collecting multi-modal data such as 2D spherical images and 3D LiDAR data represent many challenges that are commonly found in robotics research. For example, research from The Korea Advanced Institute of Science and Technology (KAIST) [29][17] found the diversity and complexity of scenes captured from multi-modal sensors, the different types of urban environment and the differences in the sensor's measurements are potential problems when processing LiDAR data for classification and segmentation. These issues from [29] can be summarised as follows:

- Inaccurate and random GPS data.
- Seasonal changes in weather and luminosity (e.g rain, fog, sunlight).
- Diversity of road conditions and road type such as multi-lane roadway.
- Complexity of the built environment and diversity of urbanisation.

A solution to tackle issues such as the diversity of the built environment is to add auxiliary labels into the dataset from other Geographic Information System (GIS) data or from the images directly. An example of the latter is UrbanLoco [62] which proposed a method for measuring urbanisation rate from street images using a Skymask [38, 62]. The example of the urbanisation rate measurement can be seen in Figure 11. The red dot indicates "the satellites blocked by high-rise structures" and green dot indicates "the satellites that are with in the line of sight" [62]. By retrieving this additional label and adapting it into the model, one can potentially improve the accuracy of the task.

Variation of Weather Generalising models to work well in different weather conditions is important. In the OSNI dataset, most of the Tracks have been recorded on a sunny or fair day during the daytime in Northern Ireland. Models trained using this data might not generalise well to different weather and lighting conditions. Examples of diverse weather condition and intensity of sunlight can be seen in the project for the Oxford RobotCar Dataset [3].

The Oxford RobotCar Dataset [3, 37] contains images and LiDAR data for over $10\ \mathrm{km}$ in central Oxford, UK[18]. The route was recorded more than 100 times covering approximately $1010.46\ \mathrm{km}$. The journey is from a human-driven vehicle, within the exact same location with:

- Differences in luminosity and sunlight (e.g. sun, dusk, night),
- Differences in weather (e.g. clouds, rain, overcast, snow),
- Differences in road conditions (e.g. poor GPS, no GPS, roadworks, detour),

[17]The KAIST dataset from [29] is available at http://irap.kaist.ac.kr/dataset

[18]The Oxford RobotCar Dataset is available at https://robotcar-dataset.robots.ox.ac.uk/

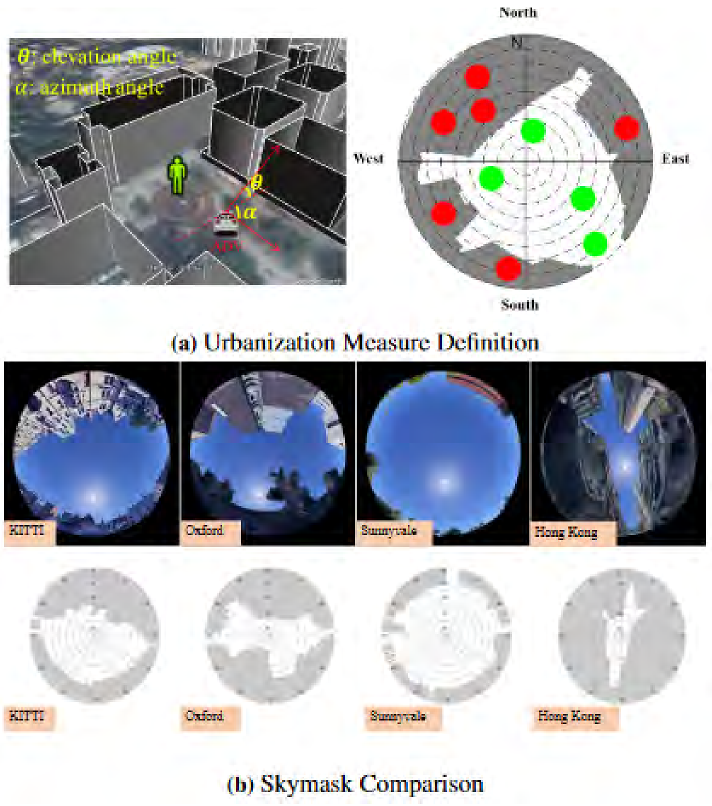


Figure 11: UrbanLoco [62] methods for measurements of urbanisation rate in a specific area by using GNSS based localisations adopted from Skymask [38, 62]

in the period of May 2014 and December 2015. outlined the proportion of the dataset from different conditions labelled on each journey. Collecting data in the future with varying weather conditions can potentially improve the performance of the model.

Data Labelling A key issue we encountered is the lack of labels in the OSNI dataset, especially in the `LiDAR` data. To alleviate this, we added additional manual labels in various formats e.g. in Chapter 5 `LiDAR` pipeline (See Figure 40). Further, we discuss an additional labelling tool for `LiDAR` (LATTE) and its extensions in Section 3.4.2.

We note that this is a wider problem. Manual data annotation via bounding boxes for `3D LiDAR` dataset is a challenging task, whether it is for an open dataset or one's own data [59]. The authors of the LATTE tool [59] discuss some of these issues with data annotation in `3D LiDAR` data (see Figure 21) which can be summarised as follows:

- Point Cloud data has a low resolution, making annotators difficult to notice or recognise the objects.
- Creating bounding boxes for `3D LiDAR` dataset is time-consuming and complex.

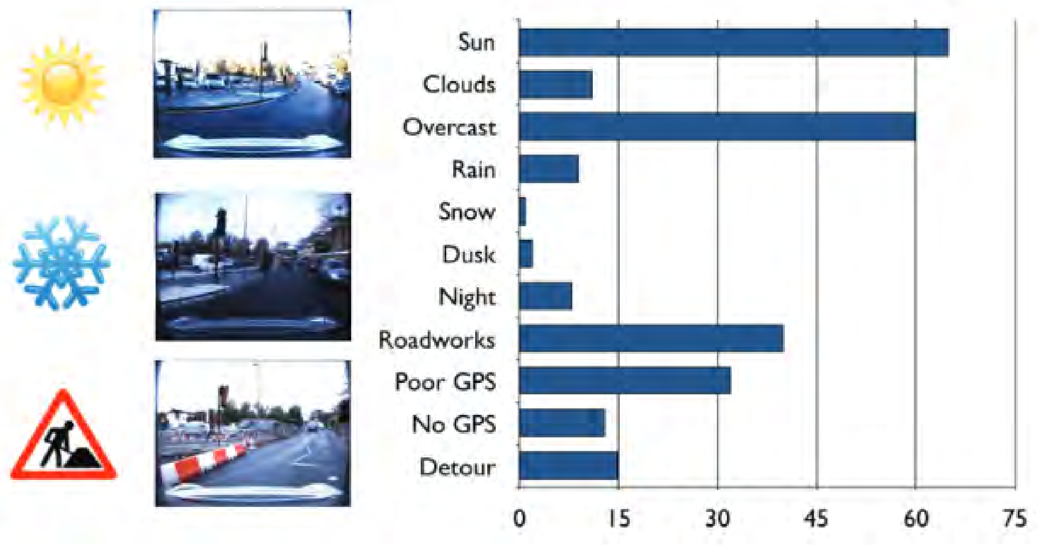


Figure 12: Various traversal conditions in the Oxford RobotCar Dataset taken from [3]

- 3D LiDAR is usually collected in a set of chronological sequence of frames.

Scalability and Computational Cost Running complex ML models can have considerable computational costs, including the requirement for Graphics Processing Unit (GPU) access to train the various models discussed in this document, several of which were trained on the smaller GPUs available in the Virtual Machine (VM) from the DSG, but some required the resources of the more powerful Joint Academic Data Science Endeavour (JADE) II cluster[19]. This should be strongly considered when taking this work forward.

[19]The JADE II cluster is a 2020 renewal of JADE which is one of the largest research GPU facilities in the UK.

3 LiDAR Data Visualisation

As mentioned in the above sections the labelling provided by OSNI was only available on the spherical images and not on the LiDAR data. Thus, it was important for us to examine these LiDAR files visually and assess the usability of these Point Clouds for the required objective of finding drains or potholes. Further, in the future the tools in this section can be used to explore each object in greater detail.

We tested multiple tools to achieve this task of Point Cloud visualisation and have documented descriptively how well they worked for us below. Most of the LiDAR files are large[20] and computational performance was an issue when working with these tools. Due to restrictions on the Microsoft Azure VM, we were not able to install and/or run some of these tools on the VM (documented below). Other considerations for the choice of tools included:

- The tools should satisfy our requirements of data security such that the data never leaves the Virtual Machine.
- The tools should be easy to install and ideally should provide an interface to Python.
- The tools should work well with the size of the files we obtained.

We will structure this section as follows. First we will discuss a well supported GUI based tool QGIS (Section 3.1), second we will discuss a common Python visualisation library called Matplotlib (Section 3.2). Finally, we consider several tools using two additional visualisation frameworks, OpenGL (Section 3.3) and WebGL (Section 3.4).

3.1 QGIS

QGIS [45] is an Open Source GIS tool that provides a desktop interface for working with GIS data. It has several tool-kits and extensions available but in our case we only explored its interaction with LAS files. The tool also allows us to overlay Open Street Map tiles [42] and other GIS modalities (if available) such as raster and vector data with the Point Cloud data. QGIS worked well with the files provided and allowed us to visualise the Point Clouds in a 2D top view fashion as can be seen in Figure 13 and Figure 14.

Installation The tool can be easily setup by downloading and installing the latest release or the long term release version from its official website[21] and is compatible with a wide array of operating systems.

[20] LAS files provided range from 18 Megabyte is 1 million bytes of data (MB) to 625 MB with a mode of 603 MB

[21] https://qgis.org/en/site/forusers/download.html

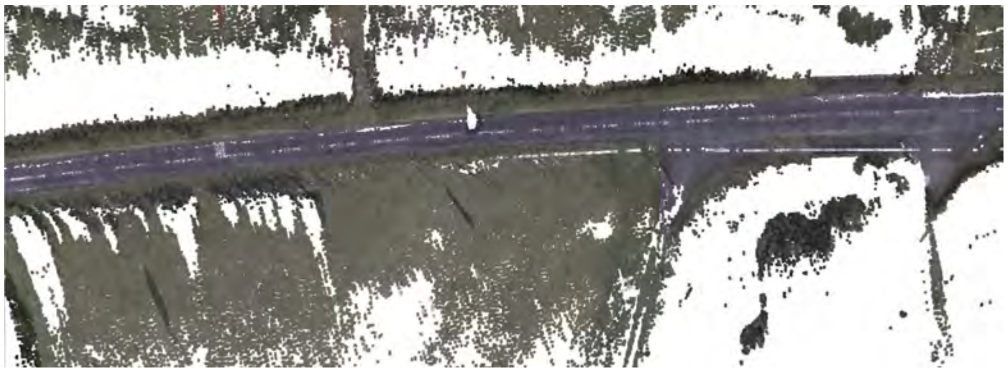


Figure 13: Example QGIS Visualisation on Track B

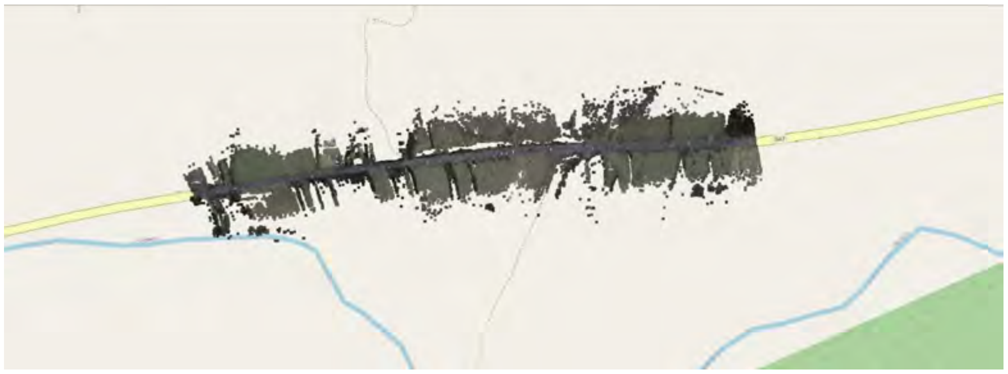


Figure 14: Example QGIS Visualisation with Open Street Map tiles on Track B from OSNI `LiDAR` Data

Limitations

- Some large files failed to display in the viewer, but the issue sometimes fixed itself on reopening the programme.
- The tool provides predominately a `2D` view of the Point Cloud. The `3D` viewer is difficult to use.

Despite its limitations, QGIS is a great tool for not only visualising Point Cloud data in a GIS platform but for analysing, joining and interacting with different types of raster and vector geospatial data. The open source platform Python interface makes it easier to build plugins and tools. As a result, there are many helpful extensions for both opening and processing `LiDAR` data. Examples for future testing include LASTools [20] and Whitebox [35].

3.2 Matplotlib

Matplotlib [27] is popular Python plotting library and it works well with `LAS` files imported using laspy [6]. The main advantage of using Matplotlib is that it is a standard Python library which requires minimal setup. However: a downside is that the plots are static. As a result, it lacks some of the interactivity and sophistication of some of the other options (Figure 15).

Installation Matplotlib can be installed in the same way as many Python packages via The Python Package Index (PyPI), and is compatible with a wide array of operating systems and Python versions.

Limitations

- Large files took a long time to visualise.
- Lack of interactivity and specialised tools to work with Point Cloud data.

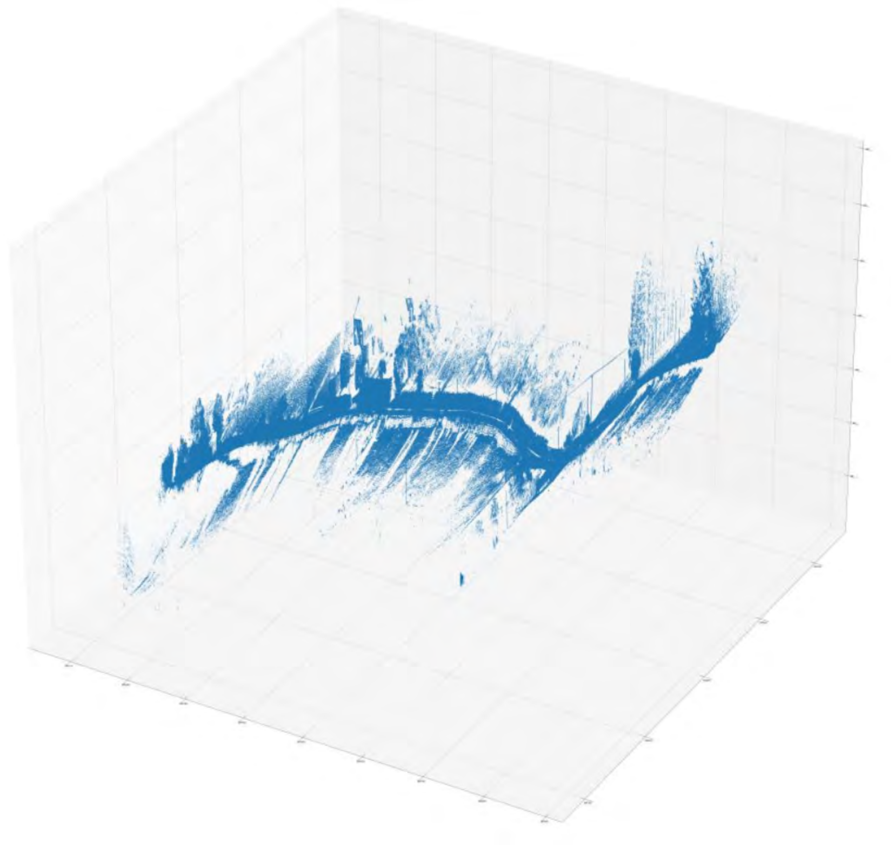


Figure 15: Example Matplotlib Visualisation on Track B

3.3 OpenGL

OpenGL [63] is an open source graphics platform for rendering 2D and 3D graphics. Due to restrictions on the graphics drivers of the VM, OpenGL tools were not installed nor extensively tested[22]. But these should work perfectly fine on a system without these restrictions which is why we think these warrant a discussion even though we were not able to run them on actual OSNI Data. In this section we will discuss two which make use of OpenGL to render and view Point Cloud data: Open3D (Section 3.3.1) and Point Processing Toolkit (pptk) (Section 3.3.2). For comparison the National (U.S.) Ecological Observatory Network (NEON) [40] dataset was used[23]. Figure 16,17 are illustrative examples of how a typical LiDAR dataset NEON can be visualised using different visualisation libraries.

3.3.1 Open3D

Open3D [68] is another library which can be interfaced directly in Python which makes it another great option for visualising Point Clouds and 3D meshes. It works using OpenGL to render the points. In terms of performance it is slower than pptk discussed in Section 3.3.2 but it does provide a lot more interactivity and tool sets for 3D ML and more.

Installation Open3D is another Python library, so you can easily install it using a package manager such as PyPI.

Limitations

- Performance above a certain threshold of points drops quickly when compared to other tools discussed here.
- Requires OpenGL and could not be tested on a DSG VM.

Figure 16 is a visualisation of a different dataset from NEON run outside the DSG VMs due to OpenGL and VM graphics restrictions.

3.3.2 pptk

pptk [36] is a Python library developed by Here Technologies Netherlands. The tool can process millions of points using an Octree like data structure[24] which makes it more

[22] Including OpenGL tools such as LASTools in QGIS.

[23] The dataset is available through an Early Analytics Course [61] developed by Earth Lab at the University of Colorado Boulder. See https://www.earthdatascience.org/courses/earth-analytics/

[24] An Octree is a tree data structure where each internal node has eight children, commonly used in Point Cloud spatial partitioning. See http://www.open3d.org/docs/latest/tutorial/geometry/octree.html for more details.

performant than other tools mentioned here (at least in terms of LiDAR data visualisation). It also provides tools to annotate points in the data. An example of the output using this tool can be found in Figure 17 using an alternative dataset.

Installation pptk can be installed via PyPI.

Limitations

- The 3D viewer is not as interactive as Open3D.
- Requires OpenGL and could not be tested on a DSG VM.

3.4 WebGL

While there is an abundance of tools that use OpenGL for visualising Point Clouds, in terms of running on the VM, the saving grace came in the form of WebGL [44] which only requires a WebGL enabled browser to work, which was available on the VM and could be easily installed. The main consideration in assessing these tools was keeping OSNI data within the DSG VMs. This was achieved by installing, running and accessing these tools locally within VMs. In the following section we discuss some important WebGL tools for LiDAR Data Visualisation namely, Plas.io (Section 3.4.1), LATTE (Section 3.4.2) and Streetscape.gl (Section 3.4.3).

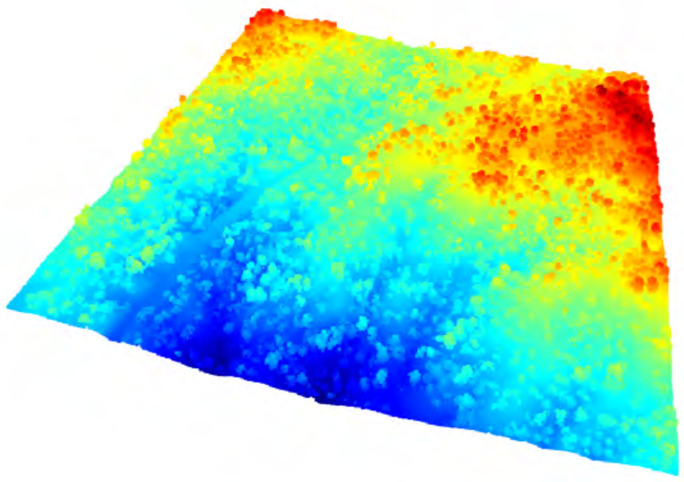


Figure 16: Example Point Cloud Visualisation using Open3D on the NEON Dataset. Illustrative example of using a different visualisation library to visualise a typical point cloud data.

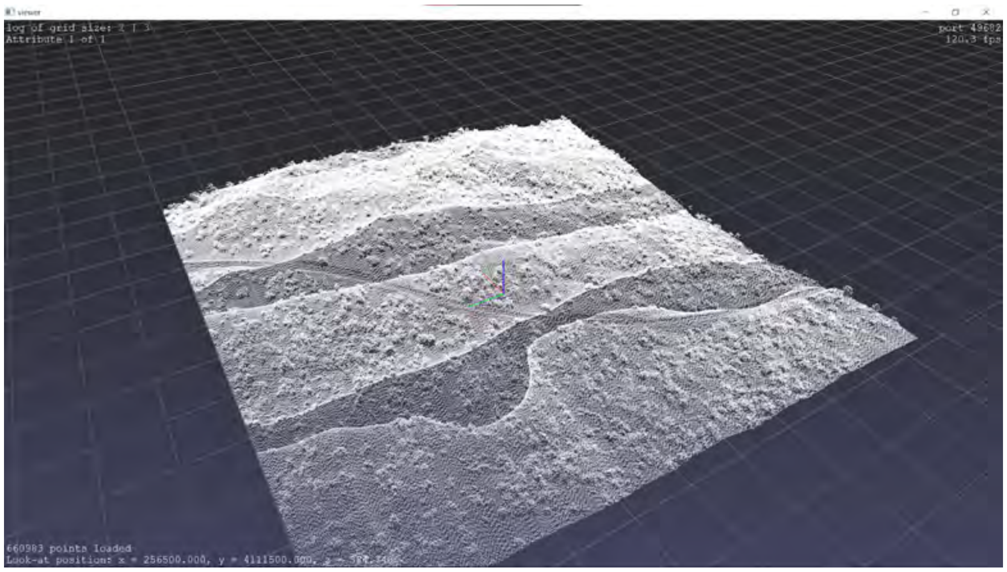


Figure 17: Example Point Cloud Visualisation using pptk on the NEON Dataset. Illustrative example of using a different visualisation library to visualise a typical point cloud data.

3.4.1 Plas.io

Plas.io [58] is a web based viewer available at https://plas.io. The tool supports LAS and LAZ files and displays them in an interactive 3D layout with features such as density reduction to improve Point Cloud loading and rendering performance. For comparison we include an example of the output on an external dataset in Figure 18 and on the OSNI dataset in Figure 19.

Installation Plas.io is available as a web app, but to ensure that the data does not leave the VM, we installed and ran a Plas.io server on one of the available VMs.

Limitations

- Can only work with LAS and LAZ files.
- The viewer is only accessible through the GUI and cannot be scripted via languages like Python.
- It is not actively maintained, and requires older versions of packages.

Plas.io was one of the only tools that worked on the VM and allowed us to visualise the LiDAR data in a 3D interactive manner. Although we cannot directly use Python with Plas.io, LAS files generated via Python can be used in Plas.io to visualise Point Cloud data generated by Python.

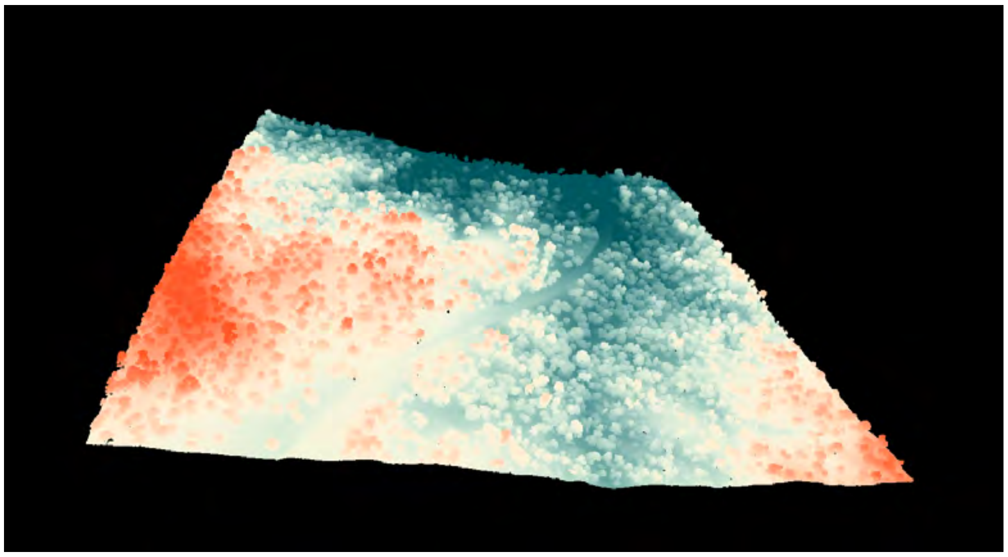


Figure 18: Example Point Cloud Visualisation using Plas.io on the NEON Dataset

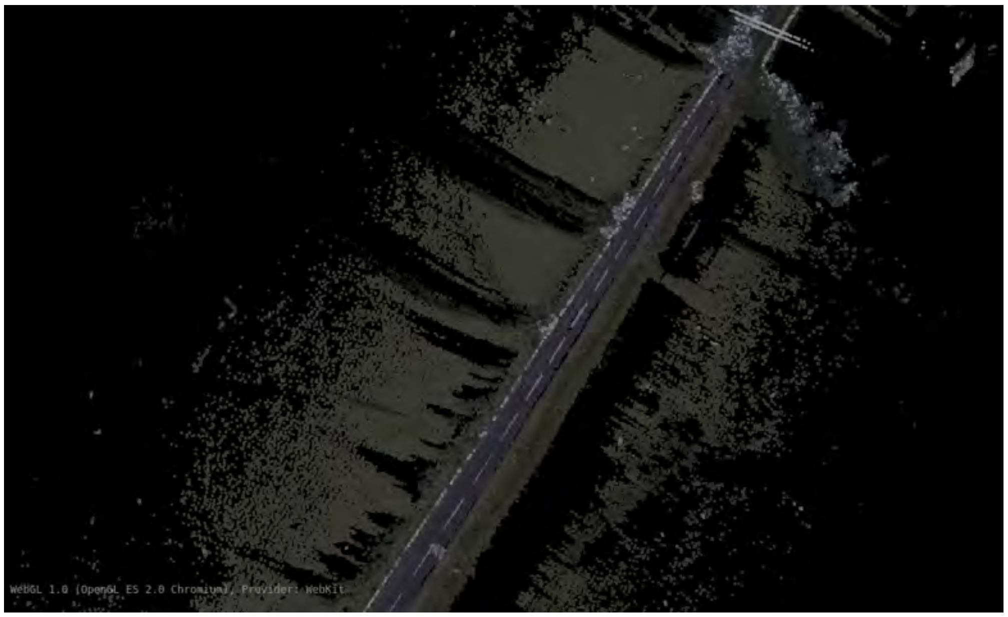
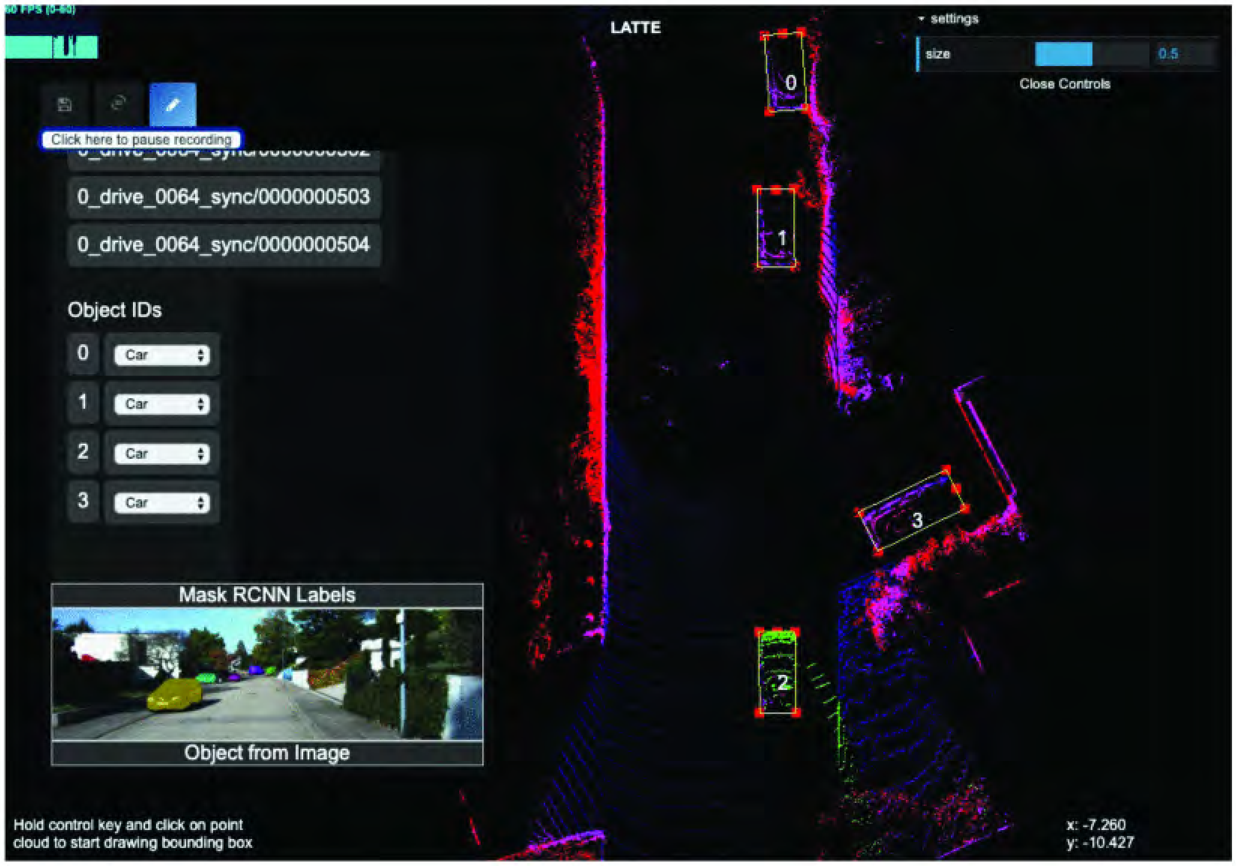


Figure 19: Example Point Cloud Visualisation using Plas.io on Track B from OSNI `LiDAR` Data

3.4.2 LATTE

LATTE is an open source project developed by Wang et. al. [59] at Berkeley AI Research, UC Berkeley. This tool allow users to manually annotate `3D LiDAR` Point Cloud dataset. LATTE proposed a "one-click annotation" tool to help annotators to draw bounding boxes for `3D LiDAR` data.

Figure 20 represents how LATTE works in drawing 3D bounding boxes of the raw data. For more information on how to annotate the data please see [59] and its official documentation.[25]



60 FPS (0-60)
LATTE
settings
size
0.5
Close Controls
Click here to pause recording
0_drive_0064_sync/0000000503
0_drive_0064_sync/0000000504
Object IDs
0 Car
1 Car
2 Car
3 Car
Mask RCNN Labels
Object from Image
Hold control key and click on point cloud to start drawing bounding box
x: -7.260
y: -10.427

Figure 20: An example of 3D LiDAR annotation by LATTE [59].

Installation The setup for the installation for LATTE goes as follows:

1. Go to LATTE GitHub project at `https://github.com/bernwang/latte`
2. Setup the environment from LATTE GitHub repositories in the local folder
3. Go to the local folder that clone the GitHub repositories and call the `app.py` file from terminal
4. Open `http://127.0.0.1:5000/` on your local browser. LATTE works in a browser that support WebGL. It is noted that Firefox has some compatibility issues.

[25]The LATTE project can be accessed at `https://github.com/bernwang/latte`

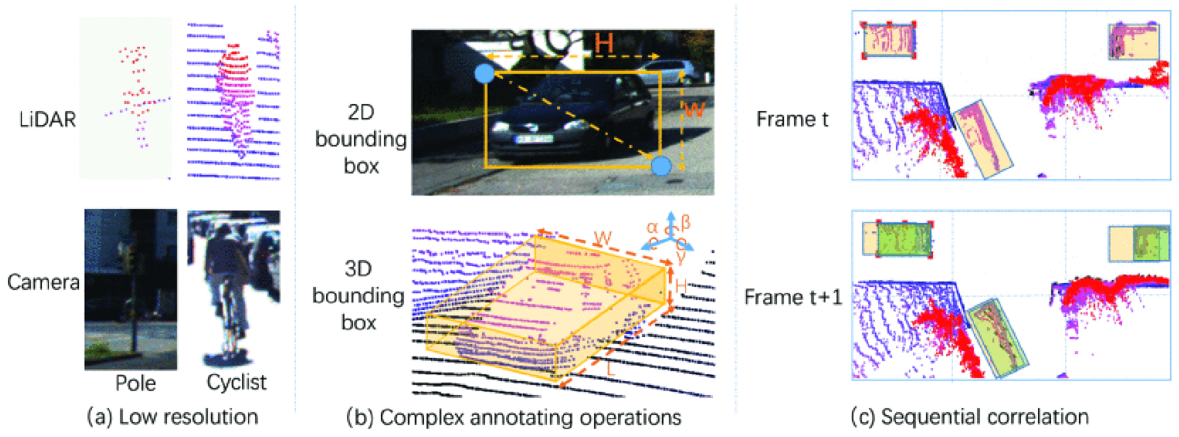


Figure 21: Challenges of 3D LiDAR data annotation Point Clouds [59]

> We tried LATTE on Microsoft Edge browser and LATTE performed well. Please turn off the network of the Edge browser: `Setting> More Tools> Developer Tools> Network> Offline`

The LATTE tool has been adopted and extended by other researchers in an open source project called Smart Annotation and Evaluation Tools for Point Cloud Data (SaNE)[26] [1]. SaNE is a semiautomatic 3D LiDAR data annotation tool for labelling Point Cloud data. In SaNE [1] they reported that: (1) they have speed up annotation process for drawing 3D bounding boxes - both for skilled and crowd sourcing human-annotators (i.e. from Amazon Mechanical Turk), and (2) reduce computational cost (compare to LATTE).

3.4.3 Streetscape.gl

Although we had minimal time testing Streetscape.gl [57], we thought it is an impressive and powerful WebGL based 3D interactive tool developed by Uber [27].

Streetscape.gl provides a declarative GUI for high quality multi-modality visualisations which are especially useful in autonomous driving applications and could be useful for visualising OSNI dataset. The tool requires data in the XVIZ format which is a protocol for real time transfer and visualisation of the data.

As mentioned from the above, we did not explore with this tool, but it has been included here as a future prospect. The site also provides a demo[28] for the toolkit on a sample of the KITTI [18] dataset. Some illustrations of the tool from the demo are shown in Figure 22.

[26]The GitHub page for SaNE is https://github.com/hasanari/sane
[27]Please see https://github.com/visgl for other tools developed by Uber
[28]The demo is available at https://avs.auto/demo/index.html

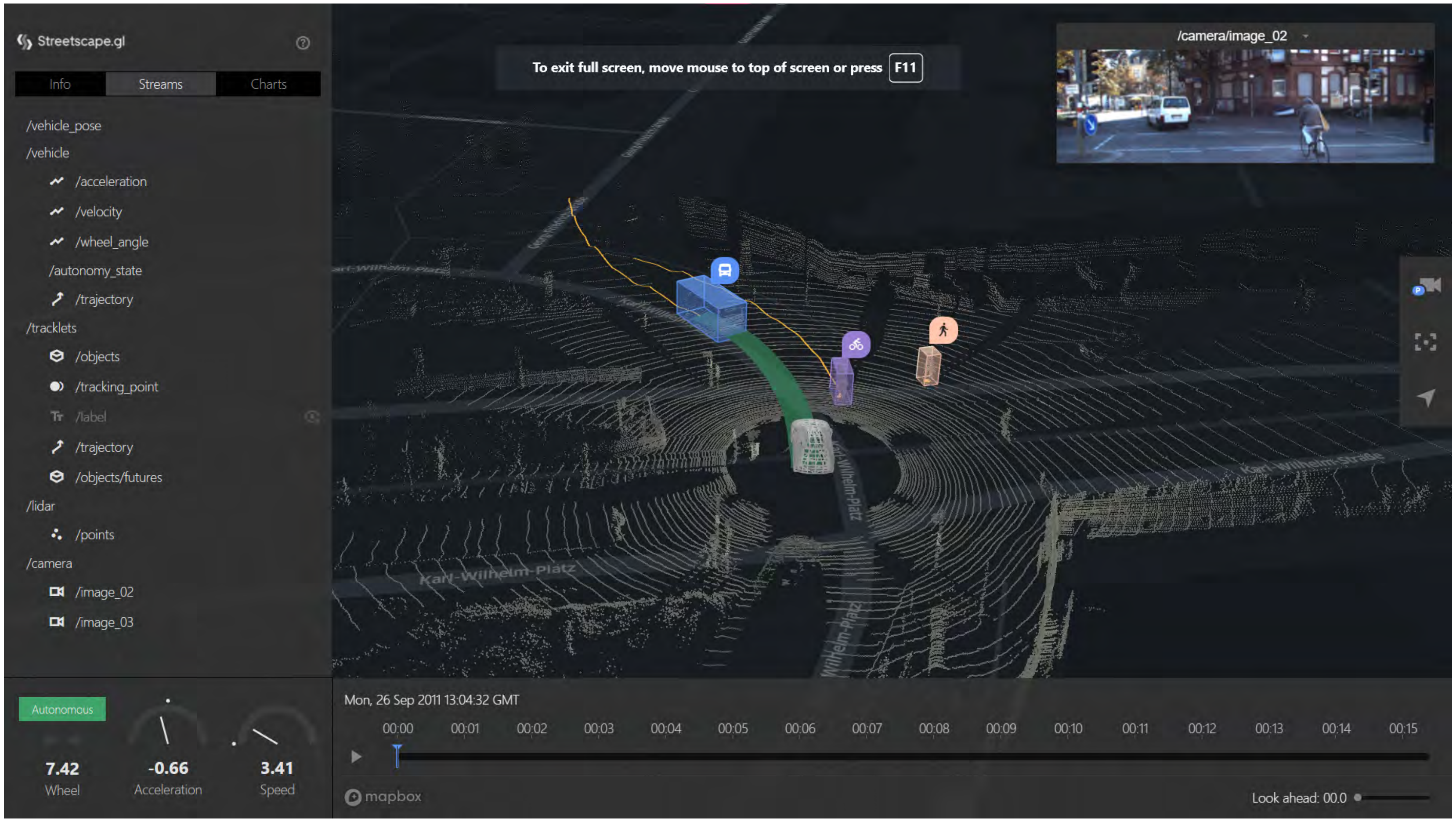


Figure 22: An screenshot from the Streetscape.gl [57] demo.

4 Spherical Imagery Pipeline

In this section we will focus on analysing the spherical imagery data and detecting drainages. We will outline several different approaches that were undertaken when analysing the `2D` spherical imagery, which we will explore in the following sections, namely, classifying by fine-tuning (modifying) a standard image classification model (Section 4.1); exploring road segmentation models (Section 4.2); using a realtime object detection model (Section 4.3) finally using a object instance segmentation model (Section 4.4).

4.1 Simple Classification Model with VGG16

We start with a simple approach where we construct a binary classifier on whether the image has a drain or not. Rather than classifying the entire image, we will focus instead on classifying sections of the images here we called Imagelets. For this approach we begin by taking a `2D` spherical image from the training set using the spatial data-split (as specified in Section 2), cropping parts of the image (given that the top part simply corresponds to the sky and the bottom is the car) and then segmenting it into a grid of Imagelets as shown in Figure 23 (left). If the centre of the bounding box provided lies within the specific grid cell, we label that Imagelet as containing a drain, otherwise marked as no drain. We repeat this for all the images in the dataset. Using this process, we end up with a highly imbalanced set of images. In order to alleviate this issue:

- We perform data augmentation for the images containing a drain. This include a left to right flip as well as extra cuts as show in Figure 23 (right) by shifting the cell by 100 pixels in 4 directions, ensuring that the centre of the drain is still included in the extra cuts.
- We then carry out a random sampling of the images that do not contain a drain, in order to have a 1:1 ratio of no-drain:drain images.

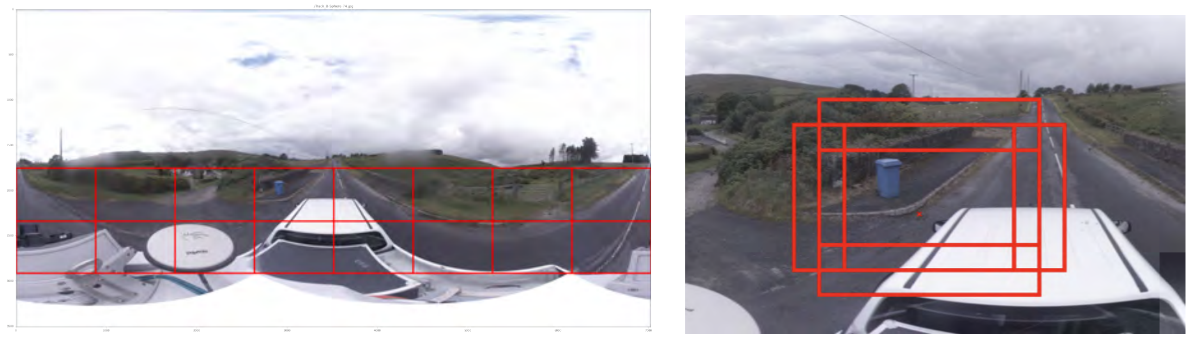


Figure 23: Example of image pre-processing for the simple classification model using an 8×2 grid. The red dot corresponds to the centre of one of the bounding boxes indicating the existence of a drain.

Given this set of images, we proceed to train our binary classifier. The model considered here is VGG16, pre-trained on ImageNet [30]. VGG16 is a standard convolutional neural network model proposed by [52] and its architecture is shown in Figure 24. The model takes an Red Green Blue (RGB) image of a fixed size as input which is then passed through a stack of convolutional blocks and fully connected layers. The fully connected layers are replaced here by a single dense layer that outputs the probabilities on whether the Imagelets contains a drain or not. Please see [52] for more information of the architecture.[29]

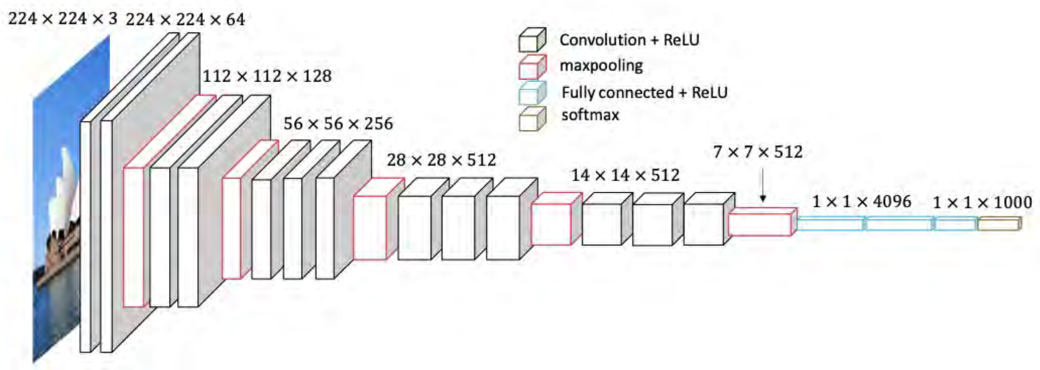


Figure 24: VGG16 architecture [52]. Figure taken from [60].

The advantage of using a pre-trained model is that it allows us to take advantage of the features that the model has learned previously, significantly reducing the training time and improving performance given the small labelled dataset.

In these experiments we have tuned our models on a validation set that was taken from the training set through an 80:20 split. Using the validation set we fine-tuned the parameters of our model (learning rate= 0.0005, momentum=0.5, epochs of training=7) as well as the size of the grid to be 15×3. Training for each epoch with the stochastic gradient descent optimiser takes approximately 30 seconds. Figure 25 shows the train and validation loss and accuracy during training.

On the test set, we neither performed augmentation nor downsampling and thus the classes are very imbalanced. To mitigate this issue when computing the confusion matrix[30] we consider an object being detected only if the model gave a probability larger than $p = 0.96$ for the spherical image, which corresponds roughly to the ratio of

[29]Note that we only added a single dense layers on top of the pre-trained model because we have limited training samples.

[30]The confusion matrix visualise the performance of the model where each row represents the counts of each labelled class and each column represents the counts of the predicted class.

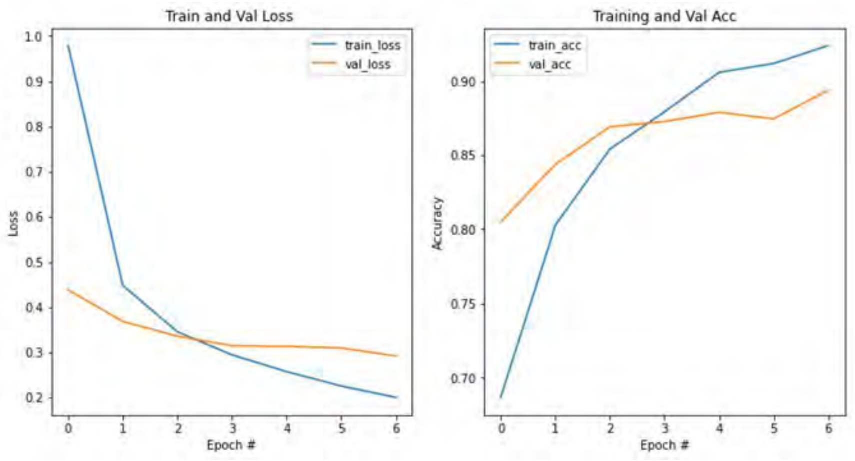


Figure 25: Train and validation loss and accuracy for the binary classifier.

no-drains/(drains+no-drains) in the train-validation set before augmentation and downsampling.

The final model achieved an accuracy of 0.977 accuracy which is marginally better than the naive classification of all images corresponding to no drains that has an accuracy of 0.965. The confusion matrices for both the validation and test set are summarised in Table 4.1 using the varying probability thresholds.[31]

The simple binary classifier on Imagelets achieved positive preliminary results. Performance on the test set can potentially improve by restricting the section of the image considered in our model even further in the dataset: currently we only cut out the sky and part of the car. This process can be combined with the masks generated from Section 4.2.

Using Grad-CAM [46, 50],[32] a popular visual explanation method for computer vision models, we visualise the areas of an image that triggers a class activation. We summarise our results in Figure 26 where the first row corresponds to true positives while the second row to false positives. We see that the algorithm appears to detect objects with striped structure or sharp intensity change, despite not being drains. The third row shows false negatives: we see that the activations are localised in the correct locations but do not carry enough weight to make a prediction. From the visual experimentation, it appears that the model is able to identify the drainage objects.

For comparison, we repeat the experiment on rectangular images, created from the spherical ones through gnomonic projections from $P = (0.75, 0), (0.75, 0.25), (0.75, 0.5),$

[31]Adjusting the probability isn't a standard protocol. As such we recommend viewing this as preliminary results. We recommend future research to use the same probability threshold on the training-validation-test set for consistency.

[32]We select the third convolutional layer as the activation layer because it appears to localise better.

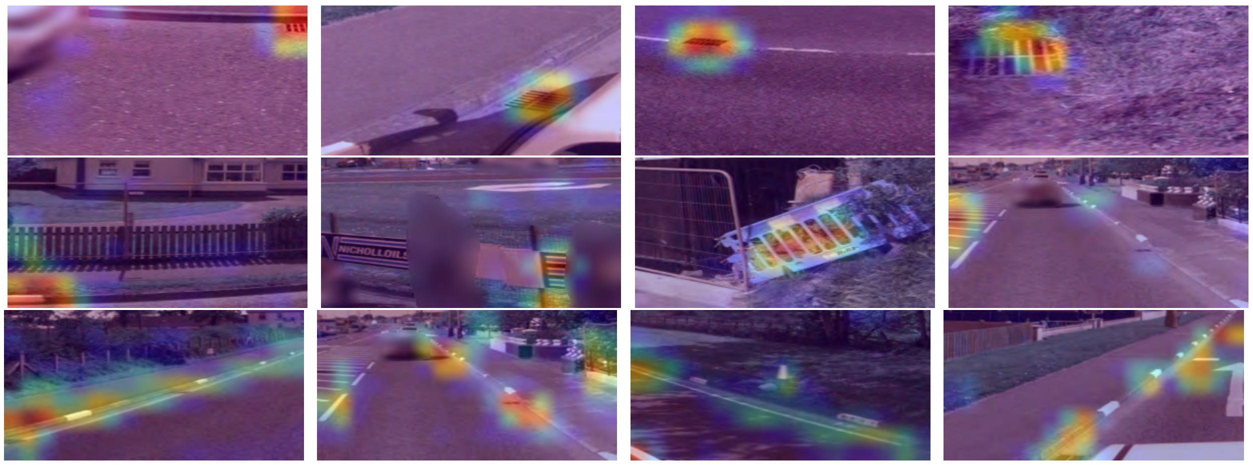


Figure 26: Activation for true positives (first row), false positives (second row) and false negatives (third row). Note that the normalisation of the activations differs across the images.

	Spherical Images	**Rectangular Images**
Val. confusion matrix	$\begin{bmatrix} 507 & 55 \\ 65 & 504 \end{bmatrix}_{p>0.5}$	$\begin{bmatrix} 438 & 184 \\ 107 & 464 \end{bmatrix}_{p>0.5}$
Test confusion matrix	$\begin{bmatrix} 3589 & 18 \\ 66 & 62 \end{bmatrix}_{p>0.96}$	$\begin{bmatrix} 3752 & 27 \\ 201 & 4 \end{bmatrix}_{p>0.85}$

Table 5: Validation and Test confusion matrices of Spherical and Rectangular images.

$(0.75, 0.75)$ and without excluding any regions from the plot. In this case we use a 4×3 grid in an attempt to maintain the same number of cropped images per spherical one.

The confusion matrices for the validation and the test set are reported in Table 4.1. We see that the analysis for the rectangular images achieves a lower accuracy than for the spherical ones, but we suspect this is because the number of cropped images was fine-tuned based on the spherical image analysis.

Overall, given the simplicity of the approach, training speed and low computational needs, the model performed well enough. It would be interesting to proceed further with a more in-depth comparison between rectangular or spherical images in order to make a concrete recommendation for OSNI to facilitate future data mining and knowledge discovery.

4.2 Road Segmentation Model

The task of manual annotation can be time-consuming, tedious and difficult. Nevertheless, large amounts of labelled images are necessary for training deep learning models and achieving good performance. Since we are dealing with high volume of spherical street images, the amount of compute and labelling effort also increases substantially.

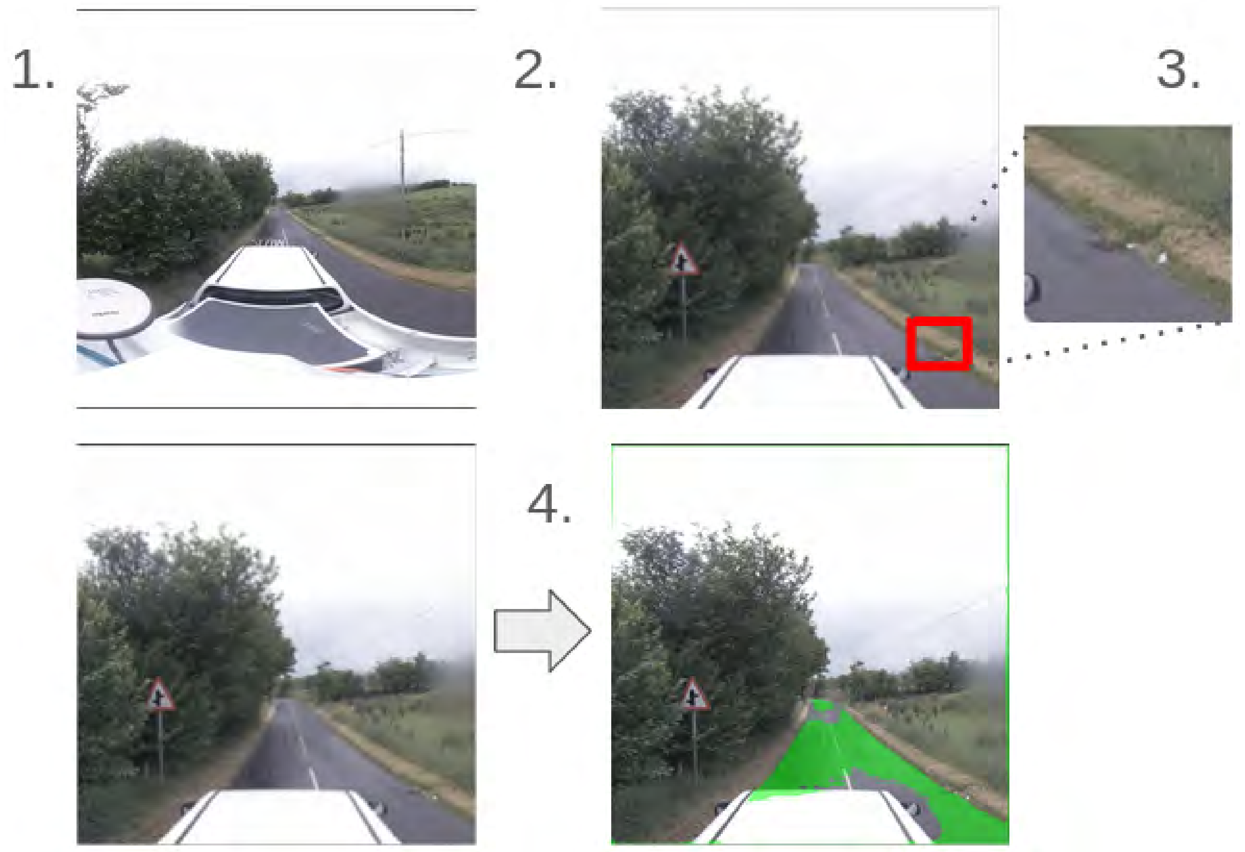


Figure 27: Outline of steps in KittiSeg road segmentation pipeline: (1) Spherical raw image, (2) conversion to rectangular image, (3) drainage location highlighted, and (4) the output of the KittiSeg road segmentation.

This presents a major challenge. One approach to tackle this is to apply pre-trained models for this specific type of images. Nowadays, there are a lot of public domain datasets such as Microsoft Common Objects in Context (COCO), PASCAL Visual Object Classes (VOC) [14] or KITTI [18]. These datasets are already annotated and contain thousands of images with different labels for different tasks, such as semantic segmentation, object detection or scene classification. By using pre-trained models many features could be retrieved without annotating a single image. The classes for e.g. road, road signs, sky, cars or person can be found in many pre-trained semantic segmentation models. To showcase the advantages of pre-trained models we applied road segmentation using the KittiSeg model [55] on OSNI data to classify the pixels corresponding to the road class.

A second advantage of using a pre-trained model is the possibility to use its outputs, in this case a mask, as part of an end-to-end model to improve accuracy. Mask detection algorithms can focus on areas where certain features might appear. For example: potholes or drains can be found within or close to the area of the road pixels.

A further advantage is it can reduce annotation time. Output from the pre-trained models can be presented to annotators for correction. This will reduce the effort for expensive feature annotation. In Figure 27 the pipeline for road segmentation using the KittiSeg model [55] can be seen, the architecture of KittiSeg is based on a Fully Convolutional Network, which is a common architecture for image semantic segmentation.[33]

This approach can be extended to other semantic classes such as sky, trees or other general features, which can be found in the public domain datasets and pre-trained models.

4.3 YOLO Model

The third approach we tested is You Only Look Once (YOLO), a real-time object detection algorithm proposed by Redmond et al. [47]. Unlike R-CNN-based models, which use multiple modules for region and class predictions, YOLO integrates these predictions into a single end-to-end neural network model. One major advantage of YOLO-based models is its speed which makes it more suitable for real-time applications.[34]

The training was completed using 570 training images from the spatial training set (as specified in Section 2).[35] The model was trained for 200 epochs where each epoch took approximately ≈ 2 minutes while the inference time per image is ≈ 0.015 seconds.

Track	**Test Intersection over Union (IoU)**
Track E	0.55
Track H	0.35
Track Q	0.31
All test	0.33

Table 6: Preliminary results on test set using YOLO with rectangular images from E, H and Q.

An example output of the YOLO detection model can be seen in Figure 28. Preliminary results are evaluated by Intersection over Union (IoU), a standard object detection metric, which are reported in Table 6. Due to the size of the drains, the out of sample results are not so bad. However, there is room for improvement, for example we can include a

[33]Please see [55] for more information on the implementation of KittiSeg.
[34]We relied on the implementation from Ge et al. for training the YOLO model. [17].
[35]The model was trained using 4 Nvidia V100 `GPUs` on the JADE II computing cluster.

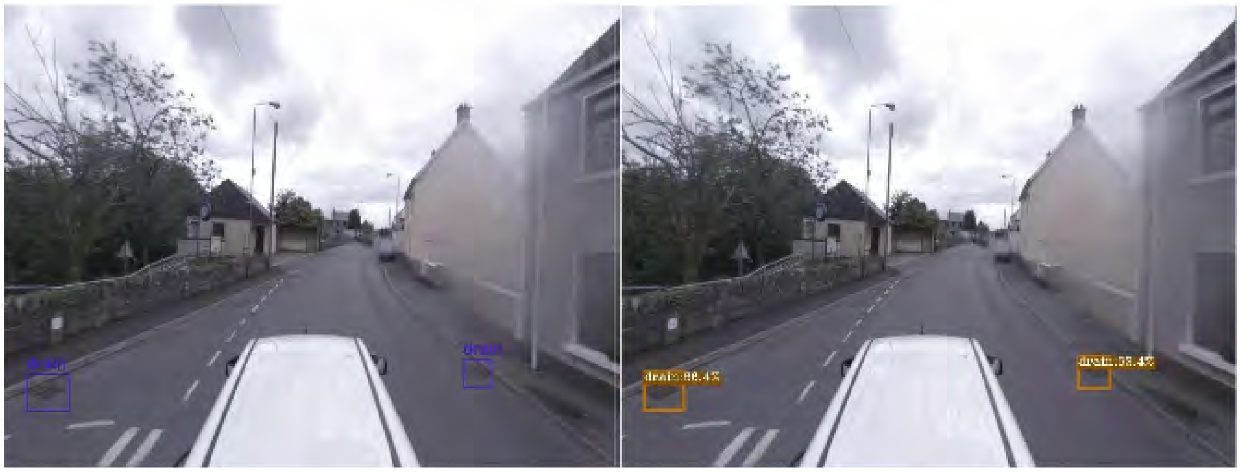


Figure 28: Example of Drainage Detection using the YOLO model. The left image includes ground truth annotations in purple and the right image is output from the detection model with the detected drainage locations.

greater number of images as training data through more varied environments or test other object detection models.

4.4 Mask R-CNN Model

The fourth approach we tested is the Mask Region Based Convolutional Neural Network (R-CNN) [24][36] model which is a framework for object instance segmentation that extends Faster R-CNN [48] by adding a branch for predicting an object mask in parallel with the existing branch for bounding box recognition. This approach was used to efficiently detect very small drainage while simultaneously generating a high-quality segmentation mask for each of them. The training was completed on 620 training images and 70 validation images that were taken randomly from the spatial training set as specified in Section 2. Following this, we evaluated the model through two different test sets where we reported the standard object detection metric Mean Average Precision [66] (mAP).

- 20 random images not in the train and the validation data.
- 10 random images each from Tracks E, H, Q (our spatial test set as specified in Section 2).

As the model required a specific data format, we reannotated the street view image data (Figure 29) into the COCO JavaScript Object Notation (`JSON`) annotation format using the

[36]Matterport's implementation of MaskR-CNN can be found at `https://github.com/matterport/Mask_RCNN`. Note: as of July 23, 2021, Matterport's version is only compatible with TensorFlow 1.15.3 and Keras 2.2.4.

VGG Image Annotator (VIA) tool [13][37].

The spherical images were processed into normal view images following the projection protocol in Section 2.2 and then resized to a uniform size of (400, 800) before annotation to make training and inference faster.

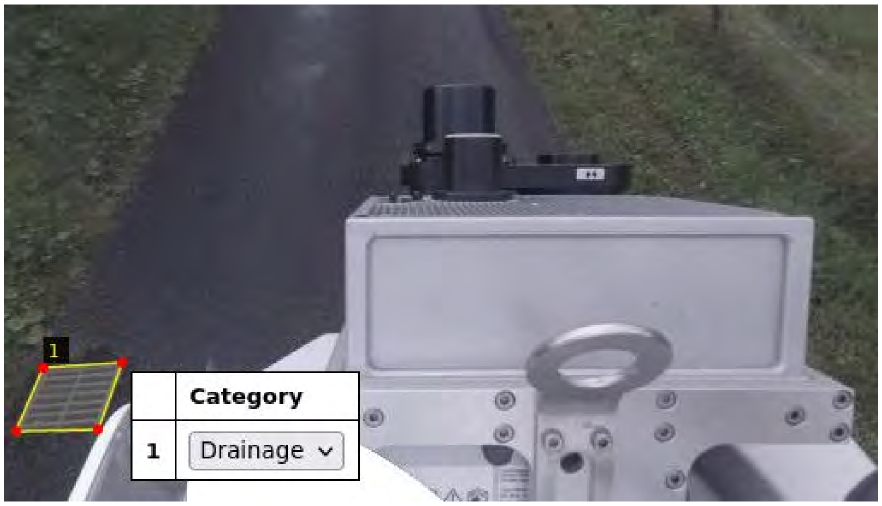


Figure 29: Polygon annotations

We will now describe in greater detail the MaskR-CNN architecture and the training procedures. The MaskR-CNN implementation here uses a ResNet50 [25] as the backbone structure, consisting of a bottom-up pathway to extract features from raw inputs, a top-bottom pathway to generate feature maps and lateral connections for intermediate operations between the two levels. This backbone structure is connected to three important stages of MaskR-CNN. In the first stage it generates proposals about the regions where there might be an object. In the second it predicts the corresponding class of the object and refines the anchors and bounding boxes. The third generates a pixel level mask of the object based on stage 1 proposals. The step by step prediction of the drainage is illustrated in Figure 30.

Training of the MaskR-CNN [24] model was performed in two stages on a Nvidia Tesla M60 GPU using the pre-trained weights from the COCO dataset. First we train the randomly initialised layers, freezing the backbone layers (pre-trained weight from COCO) for 10 epochs and later fine tune all the layers for the next 10 epochs. The learning rate

[37]The VIA tool for polygon annotations can be downloaded from https://www.robots.ox.ac.uk/~vgg/software/via The VIA tool does not require any setup or installation and it runs as an offline application in a single self-contained HTML page. It also supports manual annotation of other data modalities including audio and video. To have annotations compatible with MaskR-CNN, we exported them to the COCO JSON format. We also took advantage of polygon annotation support to make small object detection of drainage more accurate.

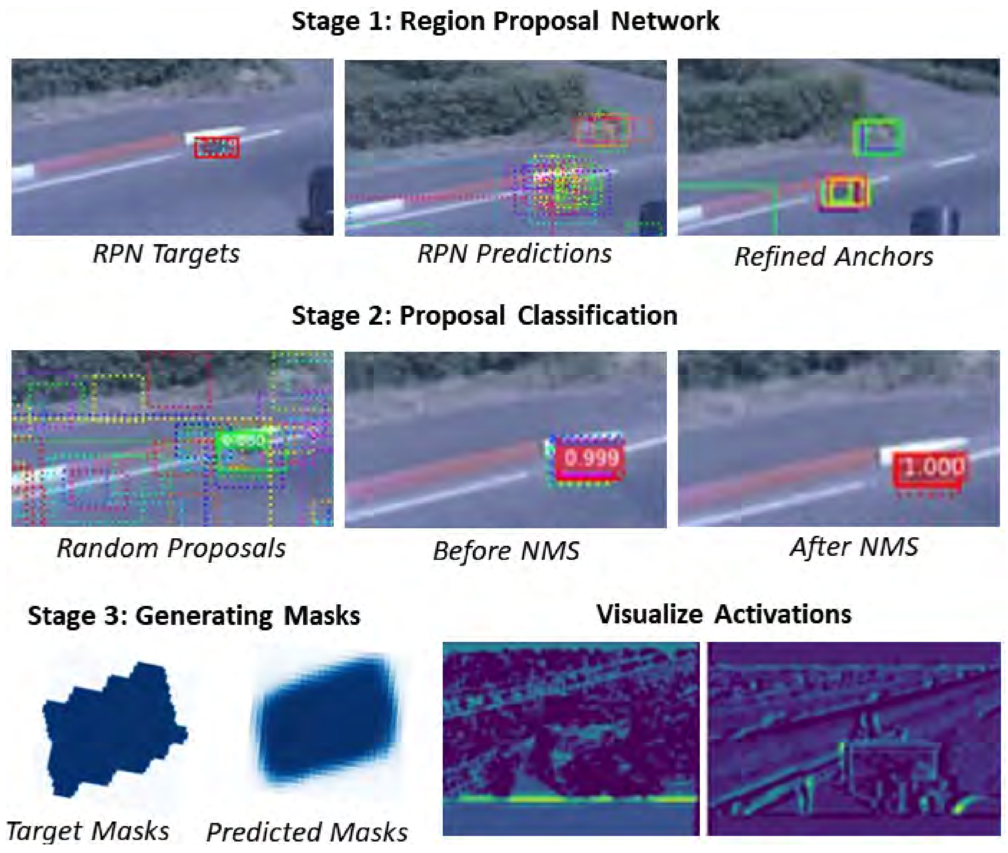


Figure 30: MaskR-CNN model pipeline. Top: proposal of regions, Middle: classification on the proposed region, Bottom: generating segmentation masks of the region and model inspections.

used for training was 0.001 and the number of classes was 2 (background and drainage).

In stage 1 of Figure 30 the Region Proposal Network (RPN) runs a lightweight binary classifier on boxes or anchors and returns scores for object or no-object. The IoU of the ground truth object (drainage) is compared with these anchors that cover the full input image at different scales for refinement. Since these anchors don't cover the drainage accurately the RPN regresses a refinement to shift and resize the anchors to the correct boundaries of the drainage based on IoU. Anchors with $IoU<0.3$ and $0.3<IoU<0.7$ are classified as negative and neutral anchors respectively and are both excluded.

Stage 2 of Figure 30 takes the region proposals from stage 1 and classifies them to generate class probabilities and filter low confidence detection and duplicates using Non-Maximum Suppression[38]. Outputs from different layers are visualised to better understand the model and to catch odd patterns. Stage 3 takes the refined bounding boxes and class

[38]A technique that optimises bounding boxes [41].

IDs and to generate segmentation masks for every instance.

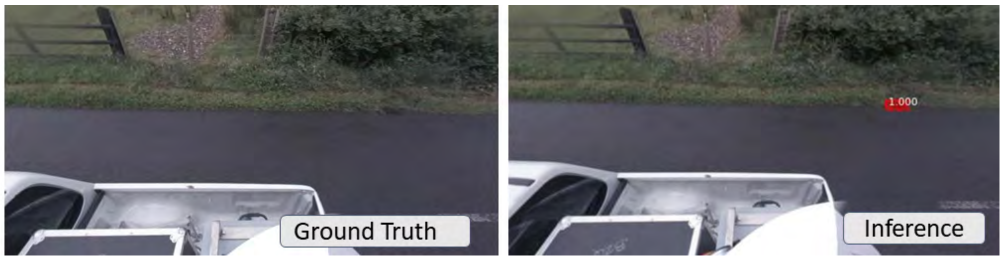


Figure 31: An image with a drainage on the left and the same image with the predicted bounding box in red on the right.

The results of the model prediction on unseen data are visualised in Figure 31. Figure 32 shows an example of some predictions made by the model and their scores. The IoU scores show a strong overlap between the ground truth bounding box and the predicted bounding box.

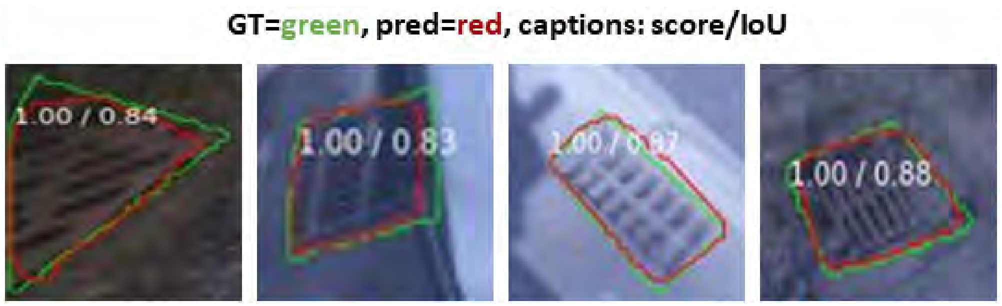


Figure 32: The Ground Truth bounding box in green and the predicted bounding box in red.

The mean IoU score computed across the first test set (20 randomly selected roads in the test set) was 0.81 while the mAP score over the same was found to be 0.67. The mAP for the second test set (40 randomly selected images selected from E, H and Q) are summarised in Table 7. Given the size of the objects and the size of the data, this result shows generally encouraging results both visually and empirically.

Track	Test mAP
Track E	0.68
Track H	0.64
Track Q	0.58

Table 7: Preliminary results on test set using MaskR-CNN with rectangular images from E, H and Q.

4.5 Summary

The results from the spherical imagery section on drainage classification and detection using different approaches are both positive and encouraging. There is room for improvements for all four approaches where increasing sample data can likely improve accuracy and model generalisation. Procedures such as incorporating pre-trained models (e.g. road segmentation models) into the pipeline can potentially improve the results. Finally, these approaches can be adapted to other urban objects detection such as potholes, benches and road quality.

5 LiDAR Pipeline

Our goal with processing the LiDAR data is to both explore the Point Cloud data and to develop a pipeline that can identify key features of the urban environment such as road drainages, road signs, potholes or street benches. There are four common tasks within a typical 3D Point Cloud workflow namely Point Cloud sampling (down-sampling to speed up computations), Point Cloud segmentation (classify each point), Point Cloud clustering (clustering points for knowledge discovery) and Point Cloud classification and fitting (fitting a classifier on a labelled dataset).

In this exploratory study, we will first describe the 3D Point Cloud attributes (eg. the Class labels) (Section 5.1), test different Point Cloud sampling strategies (Section 5.2), explore different Point Cloud clustering algorithms (Section 5.3) and finally running a Point Cloud drainage classifier with our dataset (Section 5.4). Due to the time constraint, we did not conduct any experiments on Point Cloud segmentation.

5.1 Point Cloud Attributes

We begin with describing the attributes of the 3D Point Cloud data. As shown in Table 3, the LiDAR 3D Point Cloud data provides several key attributes including X, Y, Z, Intensity, Classification, Red, Green and Blue. To explore the usefulness of the data, we visualised the 'Classification' attribute.

Figure 5.1 shows a segmentation of the inherited 'Classification' attribute. The result shows the attribute appears to contains a large number of 'unclassified' labels and a large number of overlapping classes on the road. This suggests this class attribute contained within the LiDAR dataset might not be reliable. As a result there is a need to further explore and analyse the Point Cloud data rather than relying on its existing attributes.

5.2 Point Cloud Sampling

The second task we explore is Point Cloud sampling. A major problem with 3D Point Clouds is that the data density might be more than necessary for a given application. This often leads to expensive computational cost in subsequent data processing or visualisation. To make the dense Point Clouds more manageable, data density can be reduced. Hu et al. [26] categorised existing point sampling approaches into heuristic and learning-based approaches. Since there is no standard sampling strategy, we will test two simple but efficient sub-sampling methods, namely grid sampling and decimated sampling.

Grid sampling Grid sampling is based on the division of the 3D space in regular cubic cells called Voxels. For each cell of the grid, only one representative point is kept. We perform Voxel grid sub-sampling on a Point Cloud sample. We tested this method on

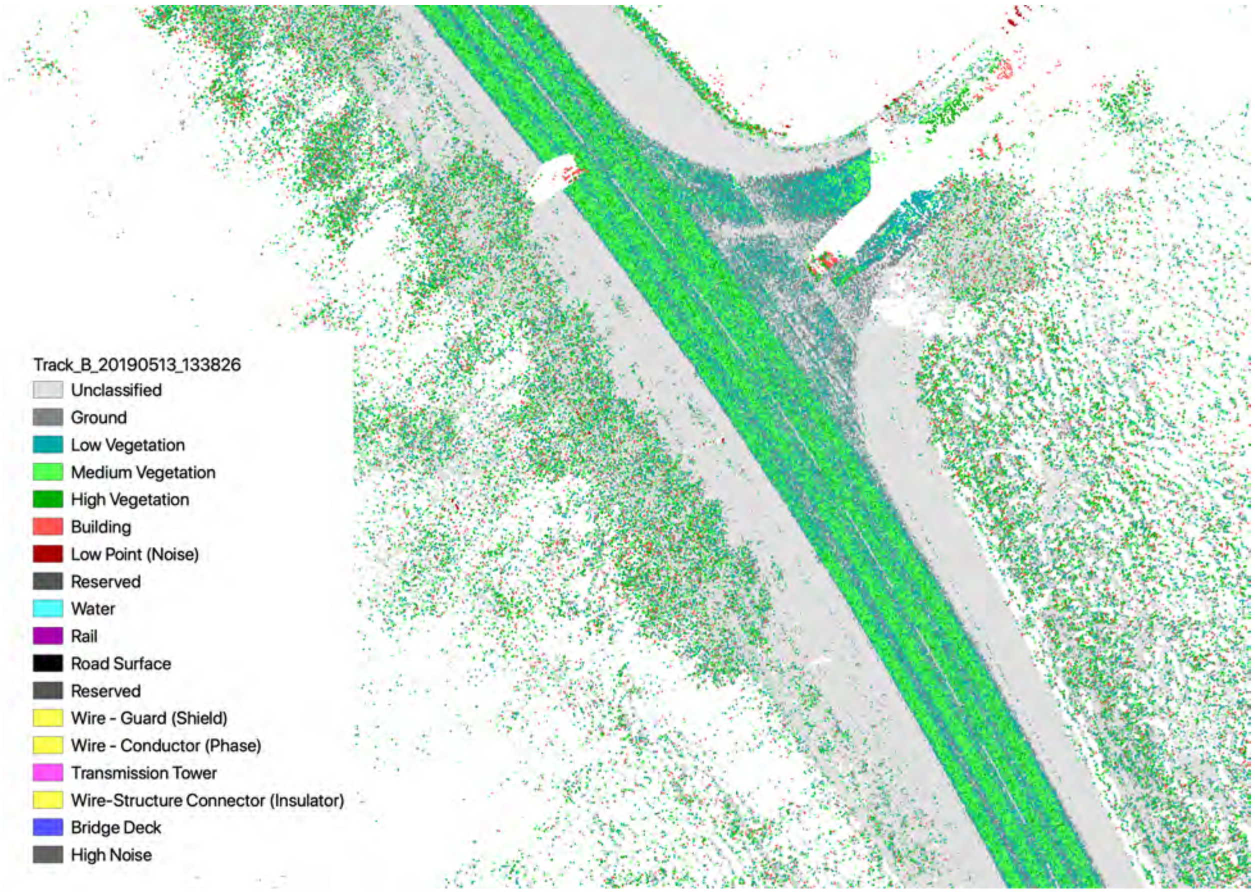


Figure 33: Segmentation based on the 'Classification' attribute contains a large number of unclassified pixels and overlapping pixels as visualised in QGIS.

a sample Track with the Voxel size set to 16. This reduces the number of points on an example file from $18,195,934$ to $11,521,396$ points (Figure 35).

Decimated sampling Alternatively there is decimated sampling which sequentially retains every n^{th} sample as seen in Figure 34. We tested this method on a sample Track with a factor set to 1600. This reduces the number of points from $18,195,934$ to $90,980$ on a sample file. The comparison of the above methods is presented in Figure 35. We found that the computational time of the above methods varies greatly. We selected decimated sampling as a pre-processing operation to reduce compute time for the Point Cloud clustering task.

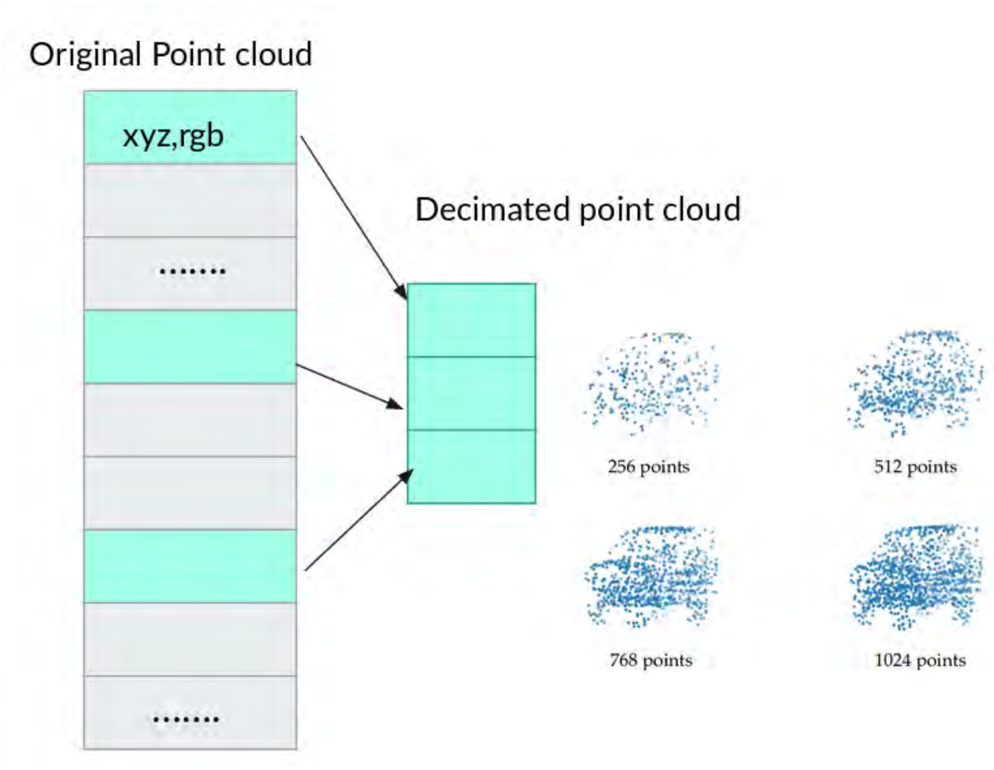


Figure 34: Decimated sampling example. [67]

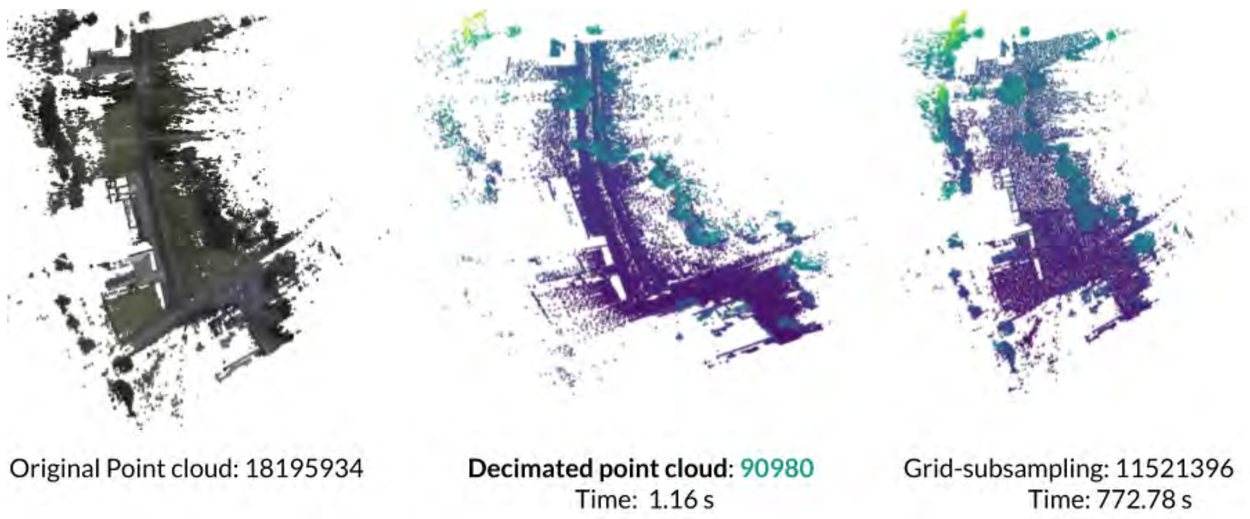

Original Point cloud: 18195934
Decimated point cloud: 90980
Time: 1.16 s
Grid-subsampling: 11521396
Time: 772.78 s

Figure 35: Point sub-sampling method comparison on a sample of the dataset. (Left) The original Point Cloud. (Centre) The decimated sampled Point Cloud. (Right) The grid sampled Point Cloud.

5.3 Point Cloud Clustering

The third task we explored in this dataset is Point Cloud clustering. Clustering analysis is an unsupervised learning method that separates the data points into several specific bunches or groups, where data points within the same groups have similar properties and data points in different groups have different properties.

As the raw `LiDAR` contained no training data, we intend to utilise standard clustering algorithms to identify local clusters with the hope the natural clusters correspond to features related to road side drainages or other urban features. We tested three algorithms, namely: KMeans, DBSCAN and MeanShift. In this case, we only used the x and y coordinates of the data for the clustering experiment. Before processing the `LiDAR` Point Cloud data, we first standarise the data to ensure that the mean for each feature was 0, and the variance was 1.

Before clustering, we also utilise the Random Sample Consensus (RANSAC) [15] algorithm to remove outliers from the decimated Point Cloud.[39] Further research is necessary to test the efficacy of the algorithm on outliers removal.

KMeans KMeans [51] is one of the simplest and most commonly used unsupervised clustering algorithms which aims to partition the Point Cloud data into k predefined clusters in which each sample belongs to the nearest cluster mean. We decided to use it as a baseline method. The model processed the `LiDAR` data to find cluster centres which are representative of various regions in the Point Cloud. Figure 36 shows the clustering result when k is set to 50. The results appears to separate the data into spatial segments of the road which doesn't relate specifically to road drainages. Further research and visualisations are necessary for clarification.

[39]RANSAC is an iterative method that estimates parameters of a mathematical model to identify inliers and outliers from the data.

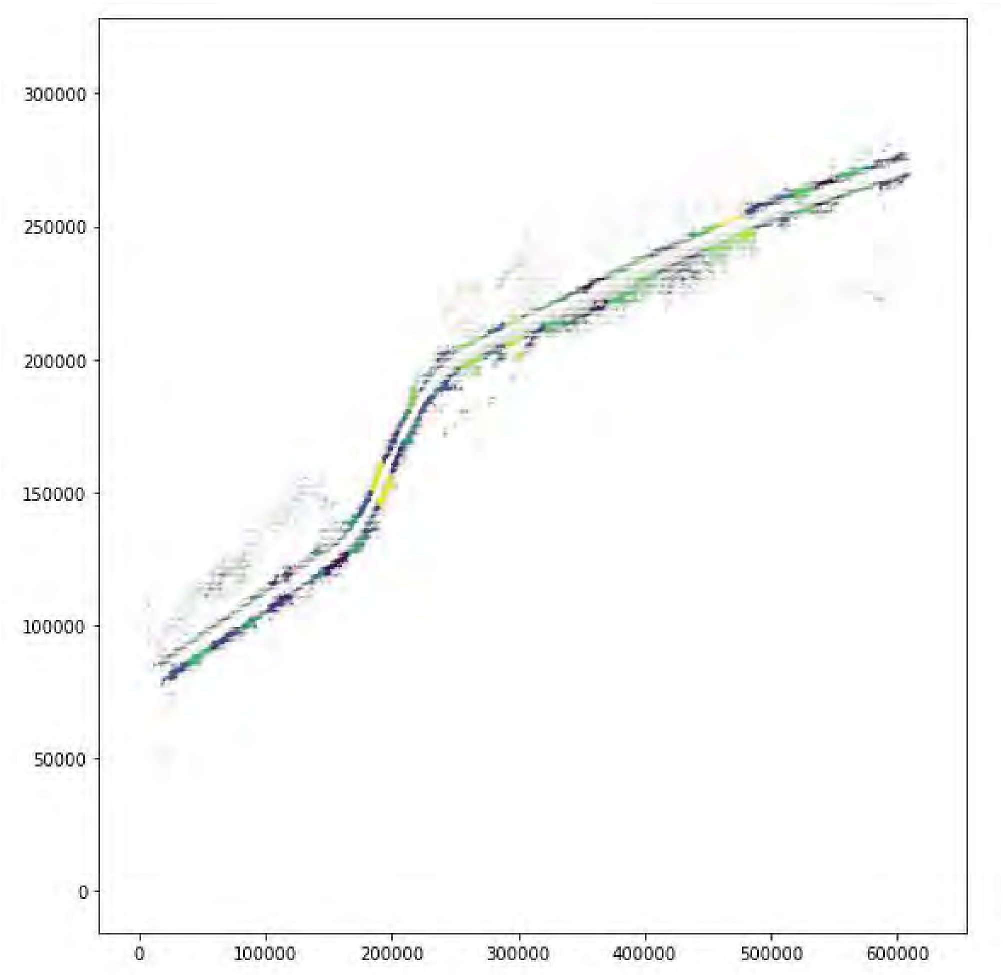


Figure 36: Plot of the KMeans model where $k = 50$ clusters. Points are coloured according to their cluster membership. These points have been rescaled so therefore the axis cannot be directly interpreted.

MeanShift MeanShift [10] is the second model that we tested. Similar to Density-Based Spatial Clustering of Applications with Noise (DBSCAN) it is an agglomerative algorithm that creates clusters iteratively, and does not require an input cluster parameter. It processed the `LiDAR` data by assigning each point to a cluster based on its distance from the centroid point. Each iteration the points get shifted to the nearest cluster. The results show this was qualitatively the worst performing model, producing unconvincing clusters. The algorithm identified two clusters while taking 258.20 seconds to execute. Figure 37 shows the plotted results from the MeanShift model.

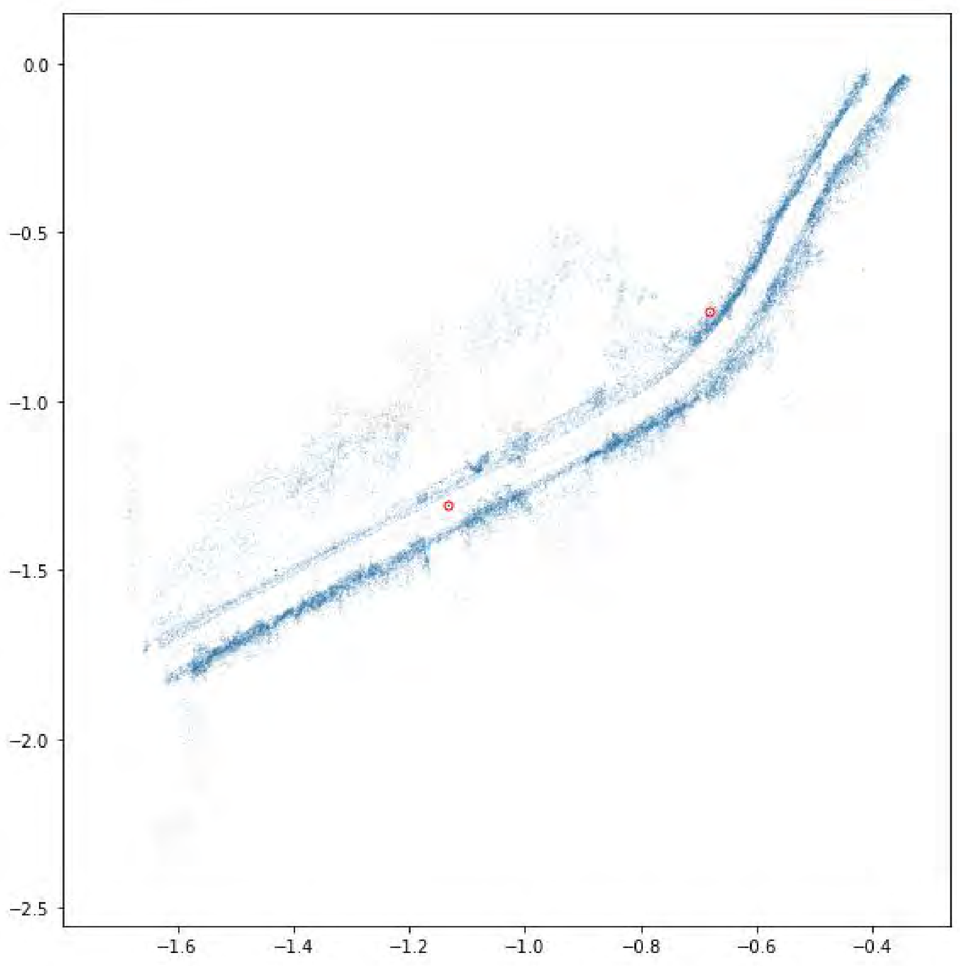


Figure 37: Plotted results of MeanShift model. Points are coloured according to their cluster membership. These points have been re-scaled so therefore the axis cannot be directly interpreted.

DBSCAN Density-Based Spatial Clustering of Applications with Noise (DBSCAN) is the third and last algorithm we tested which is another commonly used clustering algorithm to identify denser regions as clusters from the `LiDAR` data. We obtain 41 clusters by setting the Epsilon parameter as 0.1 and the minimum of samples within a cluster to 6. The results are shown in Figure 38. Comparisons varying the different parameters can be seen in Figure 39. The visual comparison shows some distinctive local clusters but more examination is require to see whether it corresponds to areas of the street which contains a road drain or other road features.

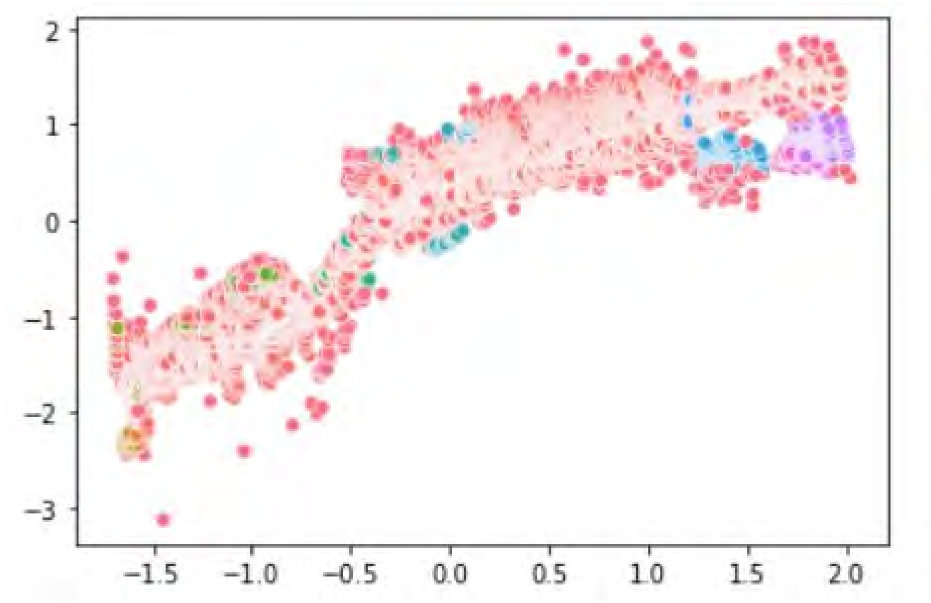


Figure 38: Plot of DBSCAN model with 41 output clusters ($\epsilon = 0.1$, min-samples $= 6$). Points are coloured according to their cluster membership. These points have been rescaled so therefore the axis cannot be directly interpreted.

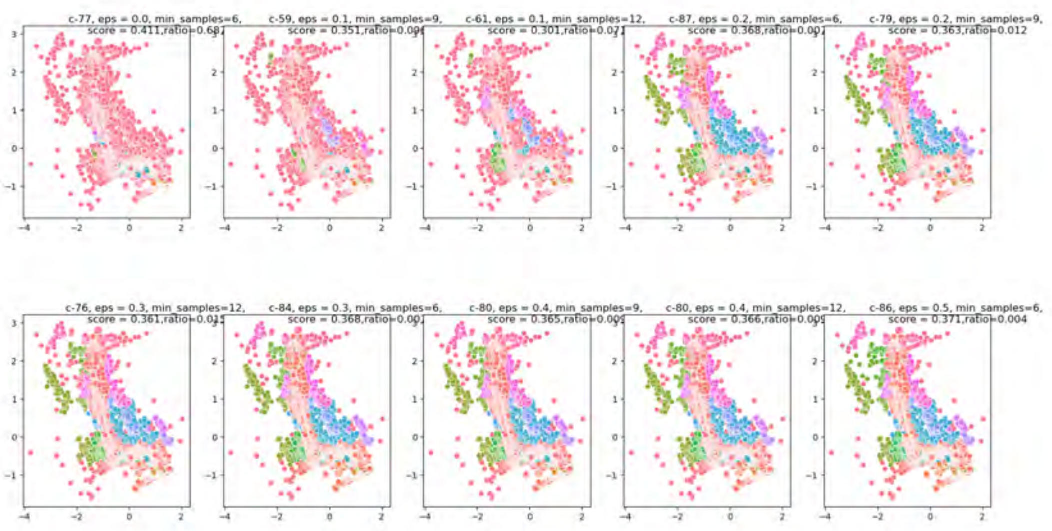


Figure 39: Visual comparison of different parameters in DBSCAN clustering. Points are coloured according to their cluster membership. Source Point Cloud from Track D. These points have been rescaled so therefore the axis cannot be directly interpreted.

Furthermore, we tested a number of evaluation metrics for this unsupervised method that can also be use for future exploration. The first metric is the noise point ratio which measures the proportion of noise in the data as found from the RANSAC algorithm where lower is better. The second metric is the adjusted-Rand Index index[40] which measures the extent the unsupervised cluster corresponds to existing ground truth class assignments where higher is better. This metric is less reliable as the ground truth classification labels is not reliable. The third metric is the silhouette score which can be used to evaluate the degree of dispersion between clusters after clustering where again higher is better.

From Table 8, we can see that the clustering labels of DBSCAN have generally low adjusted-Rand Index score and low silhouette scores. This can be attributed to both the poorly labelled data and that the cluster assignments corresponds to other objects in the urban scene. Since this is a preliminary exploration of clustering methods on raw Point Cloud data, further exploration, visualisations and tests are necessary. For example visualising the clusters labels in a 3D visualisation interactive interface can potentially offer some insights on the potential meaning of these clusters.

DBSCAN Evaluation Results

Parameters		Results			
ε	**min-sample**	**Cluster number**	**Noisy point ratio**	**Adjusted Rand Index score**	**Silhouette score**
0.05	6	69	0.299	**0.0225**	0.0295
0.1	9	30	0.143	0.014	0.0407
0.15	12	13	0.075	0.001	0.0099
0.2	6	5	0.012	0.001	0.2015
0.25	9	2	0.012	0.001	**0.2688**

Table 8: Evaluation on the DBSCAN clustering on Track D with 7 classes are shown. The highest score is highlighted in bold size.

5.4 Point Cloud Classification

In the fourth and final section, we will be constructing a classifier to detect drainage on the Point Cloud dataset, in similar fashion to the image based classifiers we constructed in Section 4.

[40]We used the following implementation https://scikit-learn.org/stable/modules/generated/sklearn.metrics.adjusted_rand_score.html#sklearn.metrics.adjusted_rand_score

Creating training data for supervised model To train a supervised model that could predict the occurrence of features, we needed to create a labelled dataset that identified the features of interest. We firstly investigated what was available on OpenDataNI which listed a number of open geospatial data in the `GeoJSON` files. Using the open 'drainage' `GeoJSON` we tried to create a spatial join with these polygons and the `LiDAR` points to identify drainage with one of our `LiDAR` files, with the prospect of using this output in our model.

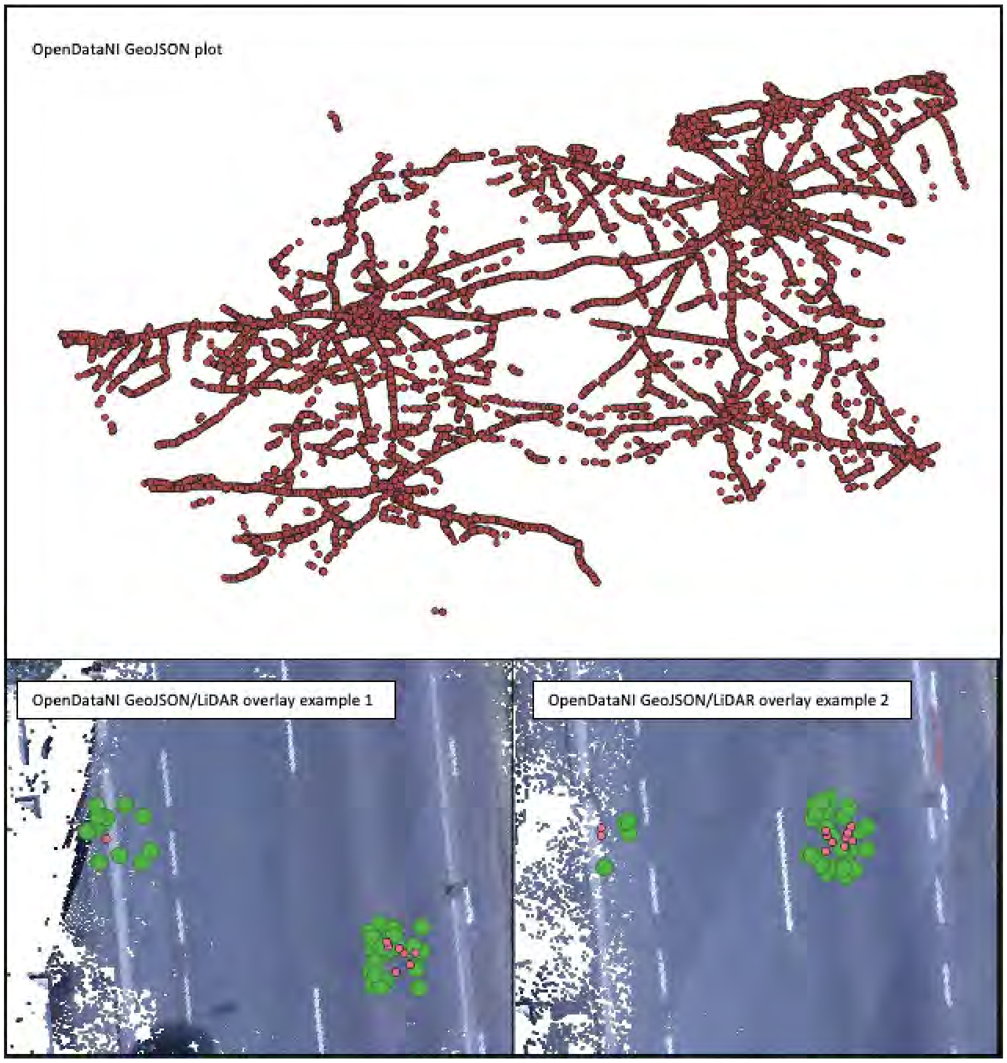

OpenDataNI GeoJSON plot
OpenDataNI GeoJSON/LiDAR overlay example 1
OpenDataNI GeoJSON/LiDAR overlay example 2

Figure 40: Image detailing the projection problem with OpenDataNI `GeoJSON` data. As seen in the lower two panels there is little or no overlap between the drains and the annotation.

We encountered a problem with this approach demonstrated in Figure 40. The `GeoJSON` polygons did not overlap with visually identifiable drains in the in the OSNI `LiDAR` data.

This was not a projection issue as the coordinate reference systems were the same, and the offset was not uniform. Therefore, this could not be fixed with a simple transform. The underlying problem is likely to be either an issue regarding the calibration of the LiDAR camera or inaccurate labelling of the open data.

To overcome this problem we manually labelled the visually identifiable drainage features, as shown in Figure 41. With such a large dataset and time constraints, we created 200 polygons[41] each for drainage and non-drainage to train the supervised model. This approach has limitations as we were only able to inspect and label a relatively small portion of the overall dataset. Also, real drain locations may not have been fully identified due to obstruction (from vegetation etc) or were unidentifiable due to sparsity (or absence) of points in particular regions of the Point Cloud. This reduces the overall accuracy and usefulness of our model.

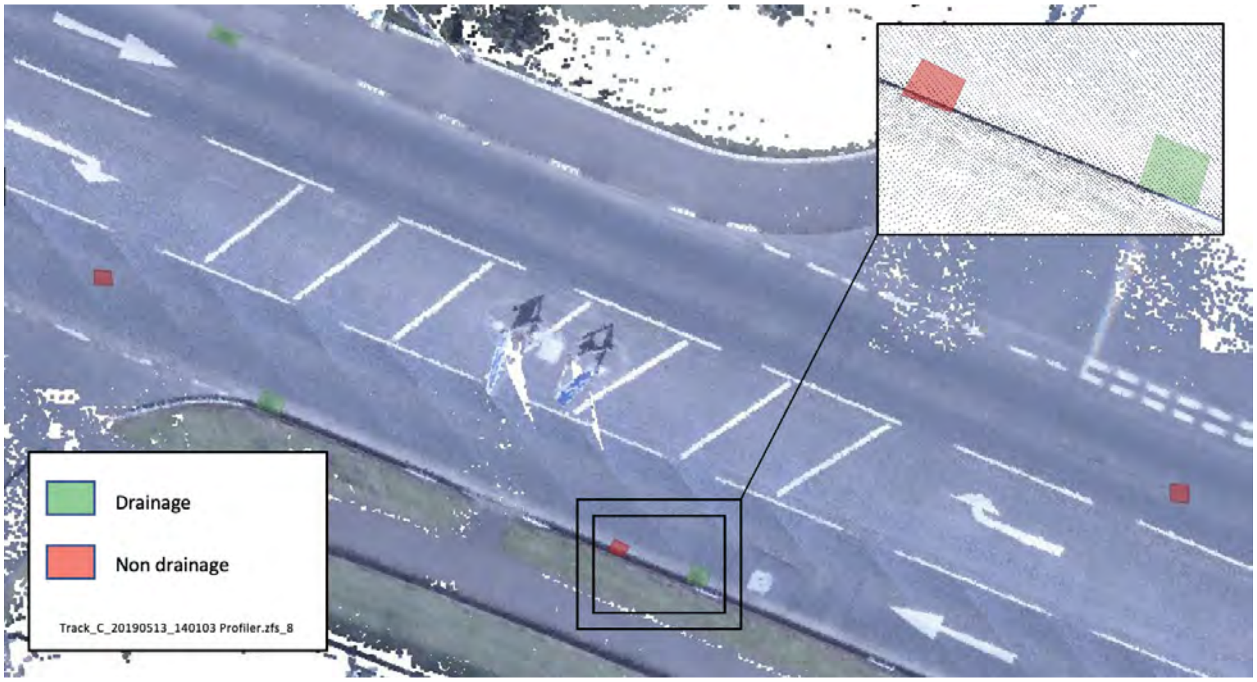


Figure 41: Manually labelled polygons for input training data

PointNet PointNet [8], which enables end-to-end learning for scattered and un-ordered point data, is one of the earliest and most popular neural network architectures for Point Cloud classification. It was specifically designed to process large 3D Point Clouds, making it ideal for our task. The model is a neural network which takes input training data to detect, classify and segment the Point Cloud. It consists of a sequence of two

[41] The assumed radius is 1m which needs to be tested in future research.

transformation networks which feed into a final feedforward neural network layer (multilayer perceptron)[42] that outputs a classification score in this case for two classes.

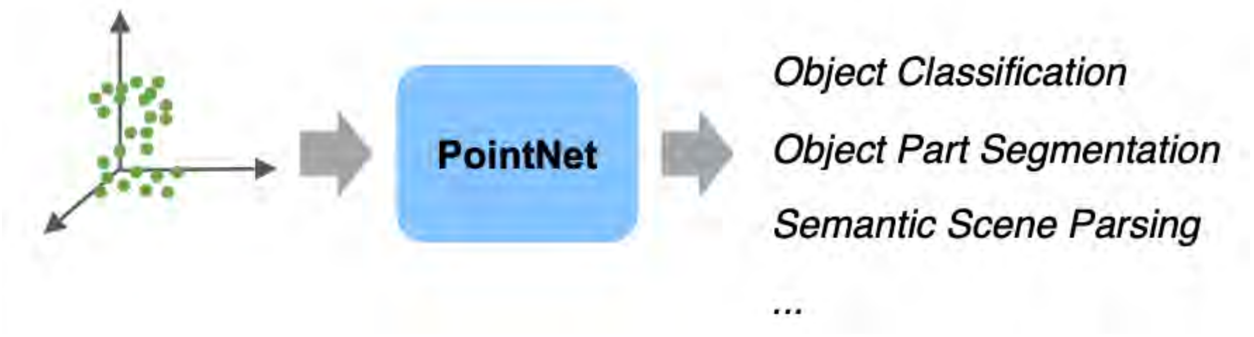


Figure 42: PointNet process diagram from Qi et al.[8]

PointNet allowed us to build a drainage classifer, but as with all neural networks it is sensitive to scaling and the choice of parameters. We utilised a PointNet implementation in TensorFlow via Keras,[43] split the annotated Point Cloud samples into a 80-20 train and test set,[44] and trained the classification model for 200 epochs.[45]

Figure 43 shows encouraging test set results with the PointNet model and Figure 44 visualises the performance of our model. Overall we obtained a test accuracy of 0.925 (with some variability between epochs although predominantly above 0.8), with a loss of 304.818 and a train accuracy of 0.978, with a loss of 0.756. The difference in the train and test results[46] are indicative of potential overfitting which could be attributed to the limited sample size. Given these constraints, this model still performed well as shown in test set accuracy and can be further improved through the use of a larger training set and more hyper-parameter tuning.

[42]Feedforward networks are artificial neural networks that have no cycles, and multilayer perceptrons are a class of these types of networks.

[43]See `https://colab.research.google.com/github/keras-team/keras-io/blob/master/examples/vision/ipynb/pointnet.ipynb`

[44]It is important to note that due to the relabelling process, this particular stream of work did not follow the spatial data split as mentioned in Section 2.

[45]We randomly sampled the Point Cloud data to 800 points, excluded data with fewer than 800 points, and set the random seed to 8.

[46]The unstable results is consistent with the original implementations.

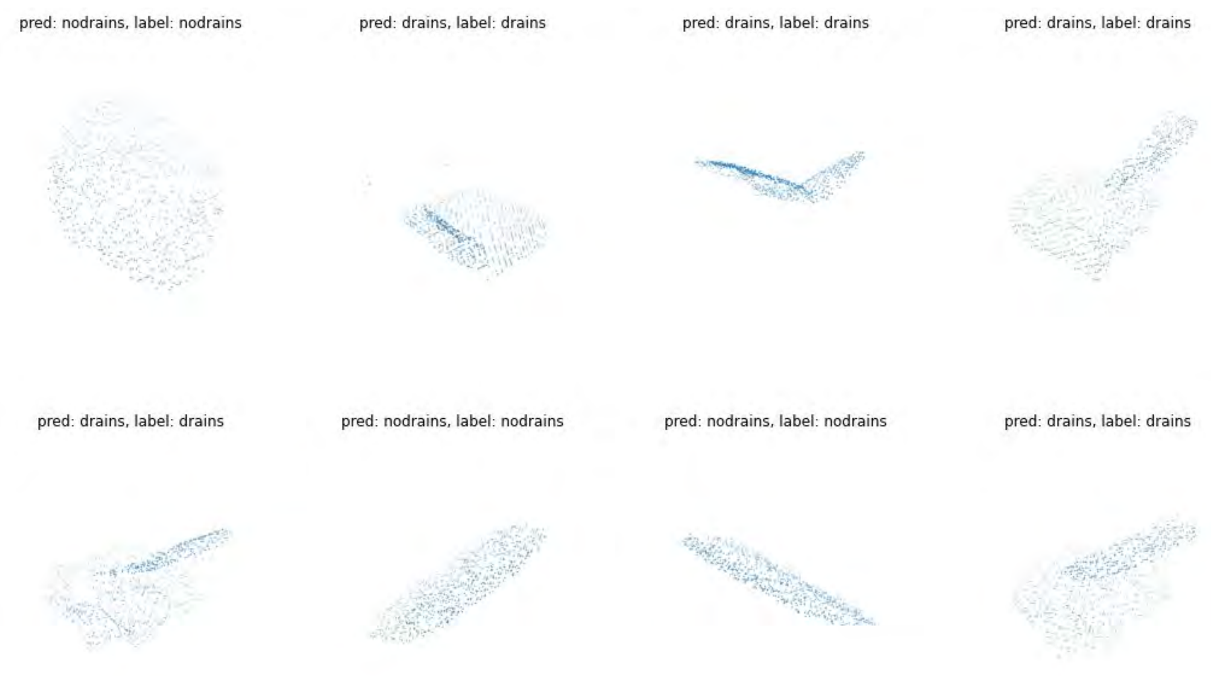

pred: nodrains, label: nodrains
pred: drains, label: drains
pred: drains, label: drains
pred: drains, label: drains
pred: drains, label: drains
pred: nodrains, label: nodrains
pred: nodrains, label: nodrains
pred: drains, label: drains

Figure 43: Plotted results of PointNet model

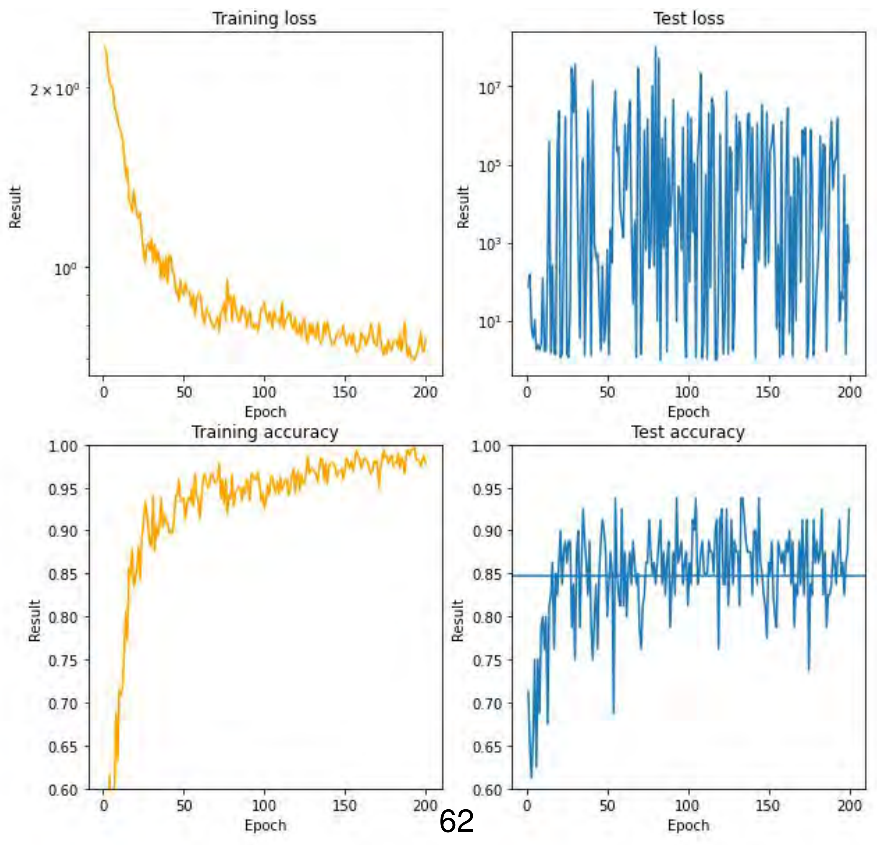


Figure 44: Performance results of PointNet model

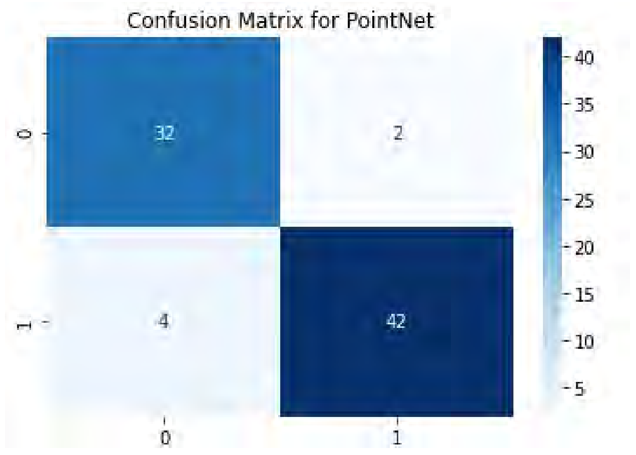


Figure 45: Confusion Matrix evaluation of PointNet model

The confusion matrix in Figure 45 summarises the performance of the model. Overall the model classified drainage (Positive) and non drainage (Negative) well with 32 True-Negatives (0,0), 4 False-Positives (0,1), 2 False-Negatives (1,0), and 42 True-Positives (1,1).

5.5 Summary

Utilising ML to process large 3D Point Clouds like LiDAR data is an active area of research. We have tested a number of tasks in the LiDAR pipeline including different types of Point Cloud sampling (e.g. grid sampling and decimated sampling), different Point Cloud clustering algorithm (e.g. KMeans, MeanShift and DBSCAN) and Point Cloud classification (e.g. PointNet).

One of the limitations of the study is we were not able to identify an effective unsupervised algorithm that helps with the drainage classification task. As such, further exploration and interactive visualisation can be utilised to derive insights from these clusters output.

We have also adapted an implementation of PointNet to tackle the drainage classification problem, where we found encouraging results. The pipeline can be replicated to identify different road features and other geographic information from the LiDAR data in the future. The main problems that we encountered is a lack of training data and the unstable test set results. The models performance could potentially be improved by increasing the sample of training data with more exploration on hyper parameter tuning and LiDAR model architecture testing.

One avenue of work which we started to investigate but did not have enough time to complete is using the PointCNN [32] architecture, which introduces a feature learning framework for Point Cloud classification and segmentation. The key component of

PointCNN is a $\mathcal{X}$-transformation which weights and permutes the input features before applying a typical convolution on the transformed features. In future work, we would like to suggest applying the PointCNN model for `LiDAR` data classification and segmentation. It is also worth noting that additional attributes like intensity, `RGB`, number of returns, etc. can potentially be used to help improve the performance of the model.

6 Discussion and Future Work

In this OSNI Data Study Group challenge, we have explored several approaches, both supervised and unsupervised, to address the issue of drainage detection. We first conducted a review of both the detection literature as seen in Section 1.3 and a review of the different visualisation tools for this data in Section 3.

Next, we have developed two pipelines: one for images and one for `LiDAR` data, and in both cases we found encouraging results. Particularly promising is the fact that it was possible to identify an object as small as a drain using either image data (VGG16, MaskR-CNN, YOLO) and `LiDAR` data (PointNet). A limitation of the study due to the time constraint is we were not able to use the same test set or same metric. As such it makes the comparison between the models difficult. This can be further expanded in future research.

It should be noted that some of these models, in order to scale up, took advantage of high performance computing services and thus OSNI may need to invest in computational resources if they want to further pursue these directions.

Future work and research avenues The range of potential applications is significant, including asset identification and management, automating identification of road sign changes for transport network and navigation data-sets, identifying feature locations such as scenic views, vegetation, drains, potholes and road surface quality, street furniture maintenance, managing autonomous vehicles, 5G mobile network planning, etc.

In terms of the pipelines developed, the following approaches might be beneficial:

- Increase the amount of labelled data to avoid over-fitting and to make the model more generalisable.
- Incorporating road segmentation into the drainage classification/detection pipeline which by leveraging this extra source of information could improve accuracy.
- Take advantage of the continuity of the images to track their locations in previous and later images. This would allow the drainage to be tracked along the road.
- Take advantage of semi-supervised approaches and active learning in order to exploit the huge amount of unlabelled imagery.
- Combine the two pipelines—`2D` imagery and `LiDAR` data—into a single multi-modal model, leveraging the advantages of both types of data.
- Utilise modern toolsets to visualise and annotate the multi-modal data.
- Incorporate transfer learning pipeline for `LiDAR` data. This would likely have improved the veracity and performance of our model.

- Test newer `LiDAR` models for processing Point Cloud datasets.

To summarise, this Data Study Group showed that there are many different opportunities for OSNI to discover new knowledge from this rich and complex source of data (eg. spherical imagery and `3D` Point Clouds captured from street-level `LiDAR` equipment). The adaptation and the deployment of methods in data science and ML offers great potential to automatically retrieve information from our environment. This information can potentially offer benefits to OSNI and beyond including many different governmental departments in urban, transport and environmental planning.

7 Team Members

In alphabetical order:

- **Mustafa Arikan** is a PhD student at the UCL Institute of Ophthalmology working on retinal diseases using computer vision, machine/deep learning for biomarkers and retinal structure detection, segmentation and on disease progression models.
- **Diego Cammarano** is a computer scientist with BSc/MSc completed at Sapienza University of Rome. He worked at the ECB and the EMA by supporting the development, deployment and maintenance of data-driven systems responsible for the processing of sensitive statistical and medical data. During his MSc thesis he worked at INGV by developing various algorithms for processing satellite images as part of a data pipeline for monitoring Italian volcanoes. He is interested in applications of AI/ML to the healthcare, earth observation and fintech sectors.
- **Ciaran Devlin** is a PhD student within the Warwick Institute for the Science of Cities (WISC) at the University of Warwick. His research interests include urban planning, urban and digital geography, spatial analysis, and design. He is a member of the Royal Town Planning Institute and a fellow of the Royal Geographical Society.
- **Andrew Elliott** (PI) is a Lecturer in the Department of Mathematics and Statistics at the University of Glasgow.
- **Vidya Ganesh** is an undergraduate student at Vellore Institute of Technology, India. She is currently interning at Ericsson Research, India, primarily focusing on Multi-agent Reinforcement Learning. She is also part of the Morse Studio Research Lab, Georgia State University, working on human psychoanalysis.
- **Irmasari Hafidz** is a second-year PhD student in Advanced VR Research Centre (AVRRC) at Wolfson School of Mechanical, Electrical and Manufacturing Engineering, Loughborough University. Her research interests are information systems, human-computer interaction and data visualisation.
- **Stephen Law** (PI) is a Lecturer in the Department of Geography at the University College London and a Turing Fellow at The Alan Turing Institute. His research interests include the application of ML in urban analytics.
- **Zhiqi Li** is a first-year PhD student in National Centre for Computer Animation at Bournemouth University. Her research interests include computer graphics, 3D computer vision, and geometry processing.
- **Sukanya Mandal** (facilitator) was the facilitator of this project. She is a Data Scientist by profession having 6+ years of experience in various domains. Her interest lies in using Data Science techniques for addressing problem statements for the greater good of the society and humans. She is passionate about utilising

data and building Artificial Intelligence (AI) systems that helps address use cases catering to UN Sustainable Development Goals.

- **Christiana Pantelidou** is currently a Marie Curie Fellow at Trinity College Dublin working in the field of Gravitational physics and String theory. She is particularly fascinated by applications of data science in the health sector, and especially in the prognosis/diagnosis of diseases, drug discovery and nutrition.
- **Yogendrasingh Pawar** is currently a final year student pursuing an MBA Tech in Finance, Computer Science from NMIMS University, Mumbai. He is interested in Fintech and Risk Management applications using Business Analytics and ML based approaches.
- **Griffith Rees** (PI) is a Post Doctoral Research Associate at the Alan Turing Institute.

Acknowledgement

We would like to thank the Alan Turing Institute and the Leeds Institute for Data Analytics as well as OSNI for their support and the computational resources provided. We are also grateful to Sukanya Mandal (project facilitator), Andrew Elliott, Stephen Law and Griffith Rees (PIs) for their help and advice on the technical aspects of the project and project management. Finally, we would like to acknowledge the use of the JADE II cluster and the University of Leeds for providing computational `hrs` on said cluster.

References

[1] H. A. A. Arief, M. Arief, G. Zhang, Z. Liu, M. Bhat, U. G. Indahl, H. Tveite, and D. Zhao. SAnE: Smart Annotation and Evaluation Tools for Point Cloud Data. *IEEE Access*, 8:131848–131858, 2020. DOI: 10.1109/ACCESS.2020.3009914.

[2] P. Babahajiani, L. Fan, J.-K. Kämäräinen, and M. Gabbouj. Urban 3d segmentation and modelling from street view images and lidar point clouds. *Machine Vision and Applications*, 28(7):679–694, 2017. ISSN: 1432-1769. DOI: 10.1007/s00138-017-0845-3. URL: https://doi.org/10.1007/s00138-017-0845-3.

[3] D. Barnes, M. Gadd, P. Murcutt, P. Newman, and I. Posner. The oxford radar robotcar dataset: a radar extension to the oxford robotcar dataset. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, Paris, 2020. URL: https://arxiv.org/abs/1909.01300.

[4] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv:2004.10934 [cs, eess]*, Apr. 2020. URL: http://arxiv.org/abs/2004.10934 (visited on 07/21/2021). arXiv: 2004.10934.

[5] N. Bowditch. NAUTICAL CHARTS. In *The American Practical Navigator*. Volume 1. 2 volumes, pages 23–50. National Imagery and Mapping Agency, Bethesda, Maryland, USA, bicentennial edition edition, 2002. ISBN: 978-1-57785-271-1. URL: https://thenauticalalmanac.com/Bowditch-%20American%20Practical%20Navigator.html.

[6] G. Brown and T. Montaigu. Laspy. URL: https://laspy.readthedocs.io/en/latest/ (visited on 06/23/2021).

[7] L. Caltagirone, M. Bellone, L. Svensson, and M. Wahde. Lidar–camera fusion for road detection using fully convolutional neural networks. *Robotics and Autonomous Systems*, 111:125–131, 2019. ISSN: 0921-8890. DOI: https://doi.org/10.1016/j.robot.2018.11.002. URL: https://www.sciencedirect.com/science/article/pii/S0921889018300496.

[8] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas. PointNet: deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 77–85, Honolulu, HI. IEEE, July 2017. ISBN: 978-1-5386-0457-1. DOI: 10.1109/CVPR.2017.16. URL: http://ieeexplore.ieee.org/document/8099499/ (visited on 07/23/2021).

[9] F. Chollet et al. Keras. https://keras.io, 2015.

[10] D. Comaniciu and P. Meer. Mean shift: a robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5):603–619, 2002.

[11] B. Coors, A. P. Condurache, and A. Geiger. Spherenet: learning spherical representations for detection and classification in omnidirectional images. In *Proceedings of the European Conference on Computer Vision (ECCV)*, Sept. 2018.

[12] T. Developers. Tensorflow, version v2.6.0-rc1, July 2021. DOI: 10.5281/zenodo.5095721. URL: https://doi.org/10.5281/zenodo.5095721.

[13] A. Dutta and A. Zisserman. The VIA annotation software for images, audio and video. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, Nice, France. ACM, 2019. ISBN: 978-1-4503-6889-6. DOI: 10.1145/3343031.3350535. URL: https://doi.org/10.1145/3343031.3350535.

[14] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2):303–338, 2010.

[15] 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*, 24(6):381–395, June 1981. ISSN: 0001-0782. DOI: 10.1145/358669.358692.

[16] J. Fritsch, T. Kühnl, and A. Geiger. A new performance measure and evaluation benchmark for road detection algorithms. In *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*. 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013), pages 1693–1700, Oct. 2013. DOI: 10.1109/ITSC.2013.6728473. ISSN: 2153-0017.

[17] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun. YOLOX: Exceeding YOLO Series in 2021. en. *arXiv preprint arXiv:2107.08430*, July 2021. URL: https://arxiv.org/abs/2107.08430v1 (visited on 07/22/2021).

[18] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun. Vision meets robotics: the KITTI dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, Sept. 1, 2013. ISSN: 0278-3649. DOI: 10.1177/0278364913491297. URL: https://doi.org/10.1177/0278364913491297 (visited on 07/20/2021). Publisher: SAGE Publications Ltd STM.

[19] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 580–587, 2014. URL: https://openaccess.thecvf.com/content_cvpr_2014/html/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.html (visited on 09/21/2021).

[20] R. GmbH. Lastools. URL: https://rapidlasso.com/lastools/ (visited on 06/23/2021).

[21] D. Griffiths. Keras documentation: point cloud classification with PointNet. URL: https://keras.io/examples/vision/pointnet/ (visited on 07/19/2021).

[22] Y. Guo, H. Wang, Q. Hu, H. Liu, L. Liu, and M. Bennamoun. Deep learning for 3d point clouds: a survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*:1–1, 2020. ISSN: 1939-3539. DOI: 10.1109/TPAMI.2020.3005434. Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence.

[23] X. Han, H. Wang, J. Lu, and C. Zhao. Road detection based on the fusion of lidar and image data. *International Journal of Advanced Robotic Systems*, 14(6):1729881417738102, 2017. DOI: 10.1177/1729881417738102. eprint: https://doi.org/10.1177/1729881417738102. URL: https://doi.org/10.1177/1729881417738102.

[24] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[25] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[26] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, and A. Markham. Randla-net: efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11108–11117, 2020.

[27] J. D. Hunter. Matplotlib: a 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. DOI: 10.1109/MCSE.2007.55.

[28] M. Isenburg. LASzip: lossless compression of LiDAR data. *Photogrammetric engineering and remote sensing*, 79(2):209–217, 2013.

[29] J. Jeong, Y. Cho, Y.-S. Shin, H. Roh, and A. Kim. Complex urban dataset with multi-level sensors from highly diverse urban environments. *The International Journal of Robotics Research*, 38(6):642–657, 2019. DOI: 10.1177/0278364919843996. eprint: https://doi.org/10.1177/0278364919843996. URL: https://doi.org/10.1177/0278364919843996.

[30] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25:1097–1105, 2012.

[31] *LAS Specification*. (1.2). Approved by ASPRS Board 09/02/2008. Apr. 2008. URL: https://www.asprs.org/a/society/committees/standards/asprs_las_format_v12.pdf.

[32] Y. Li, R. Bu, M. Sun, W. Wu, X. Di, and B. Chen. Pointcnn: convolution on x-transformed points. *Advances in neural information processing systems*, 31:820–830, 2018.

[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: common objects in context. In D. Fleet, T. Pajdla, B. Schiele, and T. Tuytelaars, editors, *Computer Vision – ECCV 2014*, pages 740–755, Cham. Springer International Publishing, 2014. ISBN: 978-3-319-10602-1.

[34] T.-Y. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014. arXiv: 1405.0312. URL: http://arxiv.org/abs/1405.0312.

[35] J. Lindsay. The whitebox geospatial analysis tools project and open-access gis. In *Proceedings of the GIS Research UK 22nd Annual Conference, The University of Glasgow*, pages 16–18, 2014.

[36] V. Lu. Pptk. URL: https://github.com/heremaps/pptk (visited on 06/23/2021).

[37] W. Maddern, G. Pascoe, C. Linegar, and P. Newman. 1 Year, 1000km: The Oxford RobotCar Dataset. *The International Journal of Robotics Research (IJRR)*, 36(1):3–15, 2017. DOI: 10 . 1177 / 0278364916679498. eprint: http : / / ijr . sagepub . com / content / early / 2016 / 11 / 28 / 0278364916679498 . full . pdf + html. URL: http://dx.doi.org/10.1177/0278364916679498.

[38] J. Marshall. Creating and viewing skyplots. *GPS Solutions 2002 6:1*, 6(1):118–120, Nov. 2002. DOI: 10 . 1007 / S10291 - 002 - 0017 - 3. URL: https://link.springer.com/article/10.1007/s10291-002-0017-3.

[39] N. Mutha. Equirectangular toolbox, 2017. URL: https://github.com/NitishMutha/equirectangular-toolbox. [Online; accessed July-2021].

[40] National Ecological Observatory Network (NEON). Discrete return lidar point cloud (dp1.30003.001). en, 2021. DOI: 10.48443/6E8K-3343. URL: https://data.neonscience.org/data-products/DP1.30003.001/RELEASE-2021.

[41] A. Neubeck and L. Van Gool. Efficient Non-Maximum Suppression. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 3, pages 850–855, Aug. 2006. DOI: 10.1109/ICPR.2006.479.

[42] OpenStreetMap contributors. Planet dump retrieved from https://planet.osm.org. https://www.openstreetmap.org, 2017.

[43] OSNI. Spatial NI - background. URL: https : / / www . spatialni . gov . uk / about-background.html (visited on 07/14/2021). [Online; accessed July-2021].

[44] T. Parisi. *WebGL: Up and Running*. O'Reilly Media, Inc., 1st edition, 2012. ISBN: 144932357X.

[45] QGIS Development Team. *QGIS Geographic Information System*. Open Source Geospatial Foundation. 2009. URL: http://qgis.osgeo.org.

[46] S.-A. Rebuffi, R. Fong, X. Ji, and A. Vedaldi. There and back again: revisiting backpropagation saliency methods. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[47] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, 2016. DOI: 10.1109/CVPR.2016.91.

[48] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. arXiv: 1506.01497. URL: http://arxiv.org/abs/1506.01497.

[49] E. Schubert, J. Sander, M. Ester, H. P. Kriegel, and X. Xu. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Trans. Database Syst.*, 42(3), July 2017. ISSN: 0362-5915. DOI: 10 . 1145 / 3068335. URL: https://doi.org/10.1145/3068335.

[50] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra. Grad-cam: visual explanations from deep networks via gradient-based localization. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 618–626, 2017. DOI: 10.1109/ICCV.2017.74.

[51] B.-Q. Shi, J. Liang, and Q. Liu. Adaptive simplification of point cloud using k-means clustering. en. *Computer-Aided Design*, 43(8):910–922, Aug. 2011. ISSN: 0010-4485. DOI: 10.1016/j.cad.2011.04.001.

[52] K. Simonyan and A. Zisserman. Very Deep Convolutional Networks for Large-Scale Image Recognition, Sept. 2014. arXiv: 1409.1556 [cs.CV].

[53] D. Steinley. Properties of the Hubert-Arable Adjusted Rand Index. *Psychological Methods*, 9(3):386–396, 2004. ISSN: 1939-1463. DOI: 10.1037/1082-989X.9.3.386.

[54] P. Sun, H. Kretzschmar, X. Dotiwalla, A. Chouard, V. Patnaik, P. Tsui, J. Guo, Y. Zhou, Y. Chai, B. Caine, V. Vasudevan, W. Han, J. Ngiam, H. Zhao, A. Timofeev, S. Ettinger, M. Krivokon, A. Gao, A. Joshi, Y. Zhang, J. Shlens, Z. Chen, and D. Anguelov. Scalability in perception for autonomous driving: waymo open dataset. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 2446–2454, Dec. 2019. URL: https://openaccess.thecvf.com/content_CVPR_2020/html/Sun_Scalability_in_Perception_for_Autonomous_Driving_Waymo_Open_Dataset_CVPR_2020_paper.html (visited on 07/15/2021).

[55] M. Teichmann, M. Weber, M. Zoellner, R. Cipolla, and R. Urtasun. Multinet: real-time joint semantic reasoning for autonomous driving. *arXiv preprint arXiv:1612.07695*, 2016.

[56] Tzutalin. LabelImg, Sept. 20, 2021. URL: https://github.com/tzutalin/labelImg (visited on 09/20/2021).

[57] Uber. Streetscape.gl. URL: https://avs.auto/index.html (visited on 06/23/2021).

[58] U. Verma. Plas.io. URL: https://github.com/verma/plasio (visited on 06/23/2021).

[59] B. Wang, V. Wu, B. Wu, and K. Keutzer. LATTE: Accelerating LiDAR Point Cloud Annotation via Sensor Fusion, One-Click Annotation, and Tracking. *2019 IEEE Intelligent Transportation Systems Conference, ITSC 2019*:265–272, Oct. 2019. DOI: 10.1109/ITSC.2019.8916980.

[60] S. Wang, Y. Wang, J. Tang, K. Shu, S. Ranganath, and H. Liu. What your images reveal: exploiting visual contents for point-of-interest recommendation. In *Proceedings of the 26th international conference on world wide web*, pages 391–400, 2017.

[61] L. Wasser. earthlab/earth-analytics-r-course: Earth Analytics Course in the R Programming Language, version r-earth-analytics, Aug. 2018. DOI: 10 . 5281 / zenodo . 1326873. URL: https://doi.org/10.5281/zenodo.1326873.

[62] W. Wen, Y. Zhou, G. Zhang, S. Fahandezh-Saadi, X. Bai, W. Zhan, M. Tomizuka, and L.-T. Hsu. UrbanLoco: a full sensor suite dataset for mapping and localization in urban scenes. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*. 2020 IEEE International Conference on Robotics and Automation (ICRA), pages 2310–2316, May 2020. DOI: 10.1109/ICRA40945.2020.9196526. ISSN: 2577-087X.

[63] M. Woo, J. Neider, T. Davis, and D. Shreiner. *OpenGL programming guide: the official guide to learning OpenGL, version 1.2*. Addison-Wesley Longman Publishing Co., Inc., 1999.

[64] R. Yang, X. Huang, and S. Chen. Efficient rendering of integral images. In *ACM SIGGRAPH 2005 Posters*, 44–es. 2005.

[65] Y.-S. Yang and Z. Koppányi. Stereo image point cloud and lidar point cloud fusion for the 3d street mapping. Mar. 2017. URL: https://www.semanticscholar.org/paper/STEREO-IMAGE-POINT-CLOUD-AND-LIDAR-POINT-CLOUD-FOR-Yang-Kopp%C3%A1nyi/7c2ea37a24b6a09266dd75936b5e4d53953d1179 (visited on 07/15/2021).

[66] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 271–278, New York, NY, USA. Association for Computing Machinery, July 2007. ISBN: 978-1-59593-597-7. DOI: 10.1145/1277741.1277790.

[67] M. Zhang, H. You, P. Kadam, S. Liu, and C.-C. J. Kuo. Pointhop: an explainable machine learning method for point cloud classification. *IEEE Transactions on Multimedia*, 22(7):1744–1755, 2020.

[68] Q.-Y. Zhou, J. Park, and V. Koltun. Open3D: A modern library for 3D data processing. *arXiv:1801.09847*, 2018.

[69] Y. Zhou and O. Tuzel. VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *arXiv:1711.06396 [cs]*, Nov. 2017. URL: http://arxiv.org/abs/1711.06396 (visited on 07/23/2021). arXiv: 1711.06396.

Ordnance
Survey