Spatial Data Analytics for Pattern Recognition of In-Situ and Wide-Area Contexts

by

Gabriel Lugo Bustillo

A thesis submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy

Department of Computing Science
University of Alberta

Abstract

Pattern recognition aims to differentiate patterns and regularities across diverse data types. Pattern recognition can identify unfamiliar objects, localize objects from various perspectives or resolutions, and infer patterns even when they are partially occluded. The ubiquity of sensors in various applications has made this topic a vital component of modern technology. Challenges encompass spatial variability, partial occlusion, data heterogeneity, dynamic environments, limited training data, noise and uncertainty, generalization issues, and performance constraints. Addressing these challenges is crucial for advancing the effectiveness of pattern recognition. ***This thesis focuses on exploring different data modalities and proposing solutions, with experimental validations, to solve pattern recognition problem for visual analysis, positioning, and pose estimation, particularly within the context of spatial and geospatial data***. Real-world scenarios, ranging from local-area in-situ sensing to wide area monitoring (WAM) remote sensing, serve as the testing grounds for this research. The proposed novel solutions tackle unique challenges posed by these domains.

In the first scenario, we focus on 3D object recognition and 6D pose estimation of texture-less objects. These are crucial and fundamental endeavors for industrial assembly line automation. While the problem of textured objects has been extensively studied, there is still an open research topic for texture-less industrial parts, which are symmetric, causing ambiguity. In this scenario, we propose a novel object localization and pose estimation technique using RGB images and depth maps of industrial assembly parts. The proposed segmentation model is fully morphologi-

cal and unsupervised for localizing the region of interest containing the target object extracted from the depth map. For object classification and pose estimation, we combine descriptors with dynamic time warping. We demonstrate how synthetic training images generated from Computer-Aided Design (CAD) models can facilitate pattern recognition.

In the second scenario, this thesis focuses on pattern recognition from 2D videos. The proposed approach is designed for the study of vehicles, with a primary focus on enhancing the assessment of goods and their value. Traditional approaches for vehicle classification often have relied on manual observation and limited sensor data, which have posed challenges in accuracy and scalability. Our proposed closed-loop system integrates deep learning and computer vision to detect, track, count, timestamp, and estimate the direction of vehicles, laying the groundwork for in-depth traffic flow analysis. The framework incorporates a unique data processing mechanism within a crowdsourcing environment, enhancing the scalability of our system. Experimental results demonstrate the effectiveness, efficiency, and robustness of the proposed system on challenging scenes, and adaptability with active learning for vehicular analysis, where the model will improve over time based on new data.

In the third scenario within the scope of WAM, our primary focus is to extract linear features from aerial images acquired by Unmanned Aerial Vehicle (UAV) for land surveying (LS). We observe that aerial photography can provide more precise geospatial and rich semantic information over large areas than conventional on-site surveying methods. However, drone captured data inherits imperfections when it is used to build and create 2D/3D maps of a physical scene. In particular, linear features are often affected in the pixel domain by factors such as complex backgrounds, different levels of occlusion, and lighting changes. In this scenario, we propose a framework for automatic surveying of road markings using drone imagery. We propose a semantic segmentation technique and a refining stage to enhance predicted masks for line connectivity.

In the fourth and fifth scenarios under wide-area remote sensing, the thesis focuses on pattern recognition of LiDAR. We introduce a novel 3D benchmark, "LiSurveying", which is a large-scale point-cloud dataset with over a billion points and uncommon urban object categories in complex outdoor environments. We propose an automatic and effective object detection and key-point feature detection pipeline on dense point-cloud scenes. The proposed method consists of a multiscale voxelization strategy to reduce the computational load and complexity of dense point clouds. Hierarchical features are then extracted to localize the objects of interest. Consequently, we propose an automatic strategy to locate the object's centroid point using a learning-based method with a space-partitioning data structure stage.

The contributions of this thesis encompass algorithm formulation. Whereas conventional pattern recognition techniques exhibit limitations preventing them from adequately addressing the aforementioned challenges, we introduce methodologies applicable in diverse scenarios.

Preface

The contents of this thesis have been published or are currently under review in peer reviewed international multidisciplinary journals and conferences. Chapter 3 describes the novel algorithm for object segmentation, detection, and pose estimation for texture-less objects in industrial environments. The outcome of this work has been published in *Array Journal (Elsevier)* and the *International Conference on Smart Multimedia (ICSM) (Springer LNCS proceedings)*. Chapter 4 discusses the details of pattern analysis from videos and suggests ways of improving model performance over time. This work has been accepted by the *IEEE International Symposium on Multimedia*. Chapter 5 introduces the first benchmark for land survey tasks and discusses feature extraction approaches on LiDAR for WAM. The results and dataset have been published in the *Symposium on 3D Object Retrieval* and *Computer & Graphics Journal (Elsevier)*. Chapter 6 describes the proposed algorithm for automatic point feature extraction and key-point estimation for automatic land survey. This work has been published in the *IEEE Conference on Multimedia Information Processing and Retrieval (MIPR)*. Chapter 7 demonstrates a novel work for linear feature extraction from aerial images. The work is under review at the *IEEE Transactions on Geoscience and Remote Sensing Journal*.

Aside from my research, I serve as a mentor and collaborator for a diverse group of 20 students spanning various academic levels and disciplines. This role includes guiding the research journeys of two PhD candidates, 11 Multimedia Master's students, two Undergraduates, one Visiting Scholar, and four participants in the Computing Science High School Internship Program (HIP) program. In this multifaceted role, I

oversee project tasks, ensuring that each student's work aligns seamlessly with our objectives. My involvement in these mentoring relationships extends beyond the classroom, as I actively engage with students in exploring cutting-edge technologies such as virtual reality, augmented reality, internet of things, and remote sensing. Together with my supervisors, I have coordinated these efforts, leveraging our collective expertise for research and innovation.

I have written this thesis in the first person plural to acknowledge and honor the contributions of my advisors and collaborators.

Acknowledgements

I would like to express my deepest gratitude to my supervisors, Professors Irene Cheng and Anup Basu, for their invaluable guidance, support, and continuous encouragement throughout my PhD program.

Second, I would like to extend my gratitude to Steve Rombough for the invaluable internship opportunities provided to me at McElhanney LTD. Working under his mentorship was an enriching experience that allowed me to participate in interesting and challenging research.

Third, I am grateful for the collaborative spirit and collective efforts of my colleagues at the University of Alberta's Multimedia Research Centre Lab.

I am also thankful to the members of my thesis committee, for their valuable suggestions and constructive criticism, which greatly contributed to the improvement of this work

I am profoundly grateful to my wife, Johana, and our daughter Emma, for their unwavering love, understanding, and unending support throughout this journey. Your patience, encouragement, and sacrifices have been my rock, enabling me to focus on my studies and overcome the challenges that came my way. This achievement is as much yours as it is mine, and I am endlessly thankful for your presence in my life.

Table of Contents

List of Tables

List of Figures

Abbreviations

6DOF Six degrees of freedom.

ALS Aerial Laser Scanning.

CAD Computer-aided design.

CV Computer Vision.

DTW Dynamic Time Warping.

FCN Fully Convolutional Network.

FPFH Fast Point Feature Histogram.

GNSS Global Navigation Satellite System.

GPU Graphics Processing Unit.

HOG Histogram of Oriented Gradient.

LiDAR Light Detection and Ranging.

LS Land Surveying.

MLS Mobile Laser Scanning.

RCNN Region-based Convolutional Neural Networks.

RoI Region of Interest.

SSIM Structural Similarity Index Metric.

TLS Terrestrial Laser Scanning.

UAV Unmanned Aerial Vehicle.

UI User Interface.

VSD Visible Surface Discrepancy.

WAM Wide Area Monitoring.

Related Publications during PhD study

- **Lugo, Gabriel**, Yair Andrade-Ambriz, Siddharth Jha, Dora Almanza-Ojeda, and Irene Cheng. *"Automatic Pavement Marking Survey from High-Resolution Drone Imagery using a Multiscale Semantic Segmentation Framework."* IEEE Transactions on Geoscience and Remote Sensing. (submitted)

- **Lugo, Gabriel**, Joey Quinlan, Lingrui Zhou, Md Nahid Sadik and Irene Cheng. *"Active Learning for Multi-Class Vehicle Categorization and Traffic Analysis in complex environments."* In 2023 IEEE International Symposium on Multimedia.

- **Lugo, Gabriel**, Rutvik Chauhan, and Irene Cheng. *"Exploring terrestrial point clouds with Google Street View for discovery and fine-grained catalog of urban objects."* In 2023 IEEE International Symposium on Multimedia.

- **Lugo, Gabriel**, Amit Upreti, and Irene Cheng. *"Multiscale point feature object localization for hydrant surveying using LiDAR."* In 2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR), pp. 372-378. IEEE, 2022.

- **Lugo, Gabriel**, Ryan Li, Rutvik Chauhan, Zihao Wang, Palak Tiwary, Utkarsh Pandey, Archi Patel, Steve Rombough, Rod Schatz, and Irene Cheng. *"LiSurveying: A high-resolution TLS-LiDAR benchmark."* Computers & Graphics 107 (2022): 116-130.

- **Lugo, Gabriel**, Nasim Hajari, and Irene Cheng. *"Semi-supervised learning approach for localization and pose estimation of texture-less objects in cluttered scenes."* Array (2022): 100247.

- Hajari, Nasim, **Lugo, Gabriel**, Harsh Sharma, and Irene Cheng. *"Markerless 3D object recognition and 6D pose estimation for homogeneous textureless objects: An rgb-d approach."* Sensors 20, no. 18 (2020): 5098.

- **Lugo, Gabriel**, Nasim Hajari, Ashley Reddy, and Irene Cheng. *"Textureless object recognition using an RGB-D sensor."* In Smart Multimedia: Second International Conference, ICSM 2019, San Diego, CA, USA, December 16–18, 2019, Revised Selected Papers 2, pp. 13-27. Springer International Publishing, 2020.

Funding Acknowledgement

1. MITACS

2. Natural Sciences and Engineering Research Council of Canada (NSERC)

3. Advanced Research and Technology program (ART) at McElhanney.

Knowledge and Technology Translation

1. Our research, "Texture-less object recognition using an RGB-D sensor," was in collaboration with industrial partner Rational Robotics, and was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) and Alberta Innovates. Rational Robotics handled the commercialization.

2. Our research, "Texture-less object recognition and pose estimation in challenge scenes," was in collaboration with Rational Robotics and was funded by NSERC and Alberta Innovates. Rational Robotics handled the commercialization.

3. Our research, "Video analysis for local-area in-situ," was financially sponsored by our industrial partner, McElhanney, and by the Mitacs Accelerate program and the Engineering Research Council of Canada (NSERC). We provided a GUI and documentation to McElhanney.

4. Our research, "UAV feature extraction for WAM," was financially sponsored by our industrial partner, McElhanney, and by the Mitacs Accelerate program. We provided the software and documentation to McElhanney.

5. Our research, "LiDAR feature extraction and object detection for WAM," was financially sponsored by our industrial partner, McElhanney, and by the Mitacs Accelerate program. We integrated our algorithm modules with Solv3D Engine for McElhanney's processing pipeline.

Chapter 1

Introduction

Pattern recognition is a discipline with significant importance across diverse domains, encompassing computer vision, speech recognition, natural language processing, and more. In computer vision, two distinct but interconnected tasks, object detection and semantic segmentation, stand out as examples of pattern recognition. Object detection involves the identification and precise localization of objects within visual data, offering applications such as autonomous driving, object tracking, and image analysis. In contrast, semantic segmentation refines pattern recognition to a pixel or point level, assigning detailed class labels to individual elements. Expanding on this foundation, the research focus of this thesis explores the field of pattern recognition across a diverse range of scales and applications. We explore five scenarios in both the local-area and wide-area domains, encompassing in-situ and remote sensing. While remote sensing entails the collection of data or information pertaining to an object, area, or phenomenon from a considerable distance, typically utilizing satellites, aircraft, drones, or other remote sensors, in-situ sensing, conversely, involves direct data collection from within the immediate proximity at specific interest locations. This fundamental distinction between these two approaches highlights the contrast, with remote sensing relying on remote data collection and in-situ sensing emphasizing direct, on-site data acquisition. Within the local-area context, stationary instances of pattern recognition in natural images are explored. Our examination encompasses tasks such as texture-less object detection and 6D pose estimation, and data analytics for comprehensive traffic pattern analysis in challenging scenarios. In respect of the wide-area domain, the research takes on the challenge of automating land survey processes, a task demanding the extraction of point and linear patterns from dense LiDAR and UAV imagery. The expansive geographical regions and broader spatial scales inherent to wide-area scenarios present unique challenges and opportunities for pattern recognition. This exploration seeks to advance our understanding

of pattern recognition techniques in the context of both fine-grained local analysis and large-scale, wide-area applications, ultimately contributing to the broader landscape of pattern recognition and its practical utility in terrestrial and remote sensing domains.

1.1 Motivation

Sensor signals can be captured in diverse dimensions and multimedia formats, ranging from trajectories and natural images to comprehensive frame collections and vast point clouds. Different sensor acquisition systems deliver real-time data with varying resolutions and complexities. Time-series signal typically refers to a sequence of data points displayed over a period of time. An image is an array of square pixels arranged in columns and rows that can appeal to visual perception in binary, grayscale or RGB. Video, as a dynamic sequence of 2D images, gains depth through additional temporal metadata. Vision sensors can provide other types of images such as depth maps, thermograms, multispectral, and hyperspectral. LiDAR emits thousands of laser pulses per second. By continuously emitting pulses and measuring their return time, LiDAR can create a detailed 3D map of the target area. However, 1D and 3D data can gather noise signals. Images are vulnerable to distortion, viewpoint variation, irrelevant background clutter, and illumination changes. Similarly, point clouds are affected by occlusion, missing points, dynamic environments, object reflectivity, and uneven point distribution. Public datasets often used in research are curated and balanced, but these ideal conditions are not always present in real-world scenarios. Processing real-world datasets poses several challenges in machine learning, especially for tasks like object detection and pose estimation. Therefore, pattern recognition plays an important role in many sensor modalities processing pipelines. Addressing these factors in pattern recognition involves a combination of preprocessing techniques, advanced algorithms for feature extraction, and recognition.

1.2 Challenges

1.2.1 Object detection and 6D pose estimation in-situ

The first problem we investigate deals with 3D object recognition and 6D pose estimation of texture-less objects in industrial scenarios. Texture-less objects in industrial scenarios are physical objects or components that lack distinctive surface textures, patterns, or color variations that are typically used for visual differentiation. Unlike

objects with clear textures, such as a striped shirt or a checkered pattern, texture-less objects have surfaces that are uniform or lack discernible features. These objects often appear smooth and monochromatic, making it challenging for computer vision systems to identify or distinguish them based on traditional texture-based methods. In industrial contexts, machines, tools, or parts used in manufacturing processes can sometimes be texture-less. Recognizing texture-less objects is a complex task in computer vision, especially in robotic automation or quality control scenarios, where precise identification and handling of industrial components are crucial. To tackle this challenge, advanced computer vision techniques often rely on shape-based recognition and depth information or employ advanced machine learning algorithms that can learn subtle geometric features to differentiate between similar-looking, texture-less objects. These methods are essential in ensuring the accuracy and efficiency of industrial automation systems. However, existing approaches normally require manual data acquisition with multiple camera view perspectives. Subsequently, they require the manual annotation of each object in each frame. This can be a rather long and tedious process when replicating in real scenarios with various texture-less objects. Apart from these obstacles, challenges arise from the object's size, shape, and material scanning, etc., which affect computer vision approaches.

Our research contributions

In this work, we have a number of new contributions compared to our previous study published in [1]:

1. We introduce a strategy for automatically generating multiple viewpoint synthetic training images from a CAD model.

2. We introduce a non-supervised clustering method for object segmentation and detection based on a depth map that does not require a training stage.

3. We demonstrate how different HOG feature parameters can be tuned to achieve better prediction and performance.

4. We develop a method for object pose estimation using Dynamic Time Warping (DTW) on natural images using the target object contour pattern and best prior candidates.

5. We show that our method achieves better time performance than other works for pose estimation.

1.2.2 Video analysis for local-area in-situ contexts

The field of traffic analysis includes challenges from the complexities of vehicular traffic dynamics in urban areas to the need for accurate assessment of goods and their value during transit. The complexities of varying lighting conditions, occlusions, and diverse vehicle shapes demand advanced algorithms, such as the integration of deep learning and computer vision techniques. Addressing challenges such as real-time vehicle detection, crowded scenarios, multiple camera views, and direction estimation requires the development of robust closed-loop systems in order to generalize models. Existing approaches for traffic analysis employing computer vision and deep learning, while promising, often grapple with limitations in generalization and scalability. These constraints are primarily revealed by the inadequacy of training data, which fails to encompass the diverse array of real-world scenarios and the absence of a continuous learning loop. Without a mechanism for ongoing learning, these models face challenges in adapting to new scenes and environments. The lack of generalization affects their usability in varying contexts. In this sense, the robustness and generalization of detection and tracking models heavily depend on the inclusivity of the training dataset, ensuring that the algorithms can adapt to the multitude of real-world scenarios encountered. A continuous learning mechanism not only refines the accuracy of vehicle detection and tracking but also equips the system to effectively navigate the challenges posed by crowded scenes and multiple camera views.

Our research contributions

- We develop a robust, closed-loop system integrating deep learning, conventional image processing, and computer vision techniques. This system enables the detection, tracking, counting, timestamping, and direction estimation of vehicles, laying the foundation for in-depth traffic flow analysis and optimization.

- We incorporate a unique data processing mechanism within a crowdsourcing environment, enhancing the scalability and adaptability of the system.

- We implement a tracking stage that computes cumulative average confidence scores per estimated class over a vehicle's lifespan, enhancing the robustness of class predictions.

- Our model can adapt through active learning techniques, indicating the system's ability to learn and improve over time, particularly in real-world vehicular analysis contexts. We achieved a 0.891 mAP score through the application of

data augmentation strategies, demonstrating the effectiveness, efficiency, and robustness of the proposed system in challenging scenarios.

1.2.3 LiDAR feature extraction for WAM

LiDAR feature extraction challenges appear due to the influence of multiple outdoor factors during 3D scanning, such as infrastructure surface characteristics (glass and mirrors), data acquisition geometry (emitted pulse energy range and angle of incidence), instrumental effects (configuration parameters such as bit depth, aperture size, amplifier, and gain control), and environmental effects (humidity, temperature, pressure, aerosol scattering, wet surfaces, and other physical variables). As a result, algorithms are widely affected by the missing point-cloud patches and interference in the data, which can easily occur in an outdoor, uncontrolled environment. Robust 3D object detection is necessary for real-world applications, such as autonomous driving and robotics. Existing LiDAR datasets include object classes such as trees, vehicles, city blocks, and a few others, which are objects mainly used in autonomous driving and urban city visualization types of applications. However, in this scenario, our research focus is land surveying and site analysis. The target object classes are very different, e.g., fire hydrants, poles, catch-basins and water-main valves. Our objective is to supplement existing datasets with these uncommon dataset objects, and support land surveying applications. It is a challenge to assess the urban assets, where they are located, and their physical condition. Therefore, we aim to obtain the spatial location information of specific urban objects on the ground, which is often composed of point and linear features. A suitable dataset is needed for land survey and engineering in order to train and validate appropriate detection and classification methods.

Our research contributions

In order to facilitate the technological advancement of LS, we introduce the high-resolution LiSurveying dataset as a benchmark, which can help to evaluate the performance of classification, segmentation, and detection algorithms. Our main contributions are summarized below.

1. We present the first large-scale hybrid (point-wise and 3D bounding box) TLS-LiDAR dataset for LS. Our dataset provides 360 degrees of sensor coverage in multiple locations of the scene, where multiple scans are integrated to obtain a massive, point-cloud representation. The representation is validated with

accurate CAD drawings, surface models, and latitude and longitude metadata information of each scene collected by the surveyors.

2. LiSurveying provides rich data for research on object classification, semantic segmentation, and object detection for complex outdoor scenes.

3. We compare the performance of our model using hand-crafted features against deep learning models for 3D object classification.

4. We implement a multiscale dynamic voxelization pipeline to split the raw input point-cloud into fixed voxels based on prior information from the training data.

5. We propose an object detection model that learns hierarchical features with increasing scales at each interest spatial voxel.

6. We extend our model with a fast key-point detection based on hierarchical features extracted from KD-trees for centroid point estimation.

1.2.4 UAV feature extraction for WAM

The land surveying and planning communities are interested in the accurate surveying of road paint line features from aerial images because these systems provide more precise geospatial and rich semantic information over a large rural area than conventional on-site surveying methods. The essential role of surveying is to build and create 2D/3D maps of a physical scene. Nevertheless, linear features are often affected in the pixel domain by factors such as complex backgrounds, different levels of occlusion, missing pixel information, and lighting changes. In addition, roads and painted lines represent a small portion of the pixels with respect to the extension of the image, and this usually leads to the imbalance problem associated with these classes. Furthermore, many challenges still exist for robust, fast, and accurate vision-based linear feature extraction, despite the fact that several SOTA methods have proven successful in scene understanding and remote sensing. These methods continue to have insufficient capabilities for retrieving local contextual features, which affects segmentation accuracy.

Our research contributions

Our main contributions are summarized below.

- We present a framework for automatic surveying of painted traffic lines on rural scenarios for two classes "center-line" and "edge-line," from high resolution

aerial images. Our proposed method is mainly based on semantic segmentation models to locate the patterns of lines embodied in the road surface. Subsequently, the predicted masks with the different per pixel classes are post-processed in a cascading module to remove irrelevant artifacts in neighboring pixels. Finally, a refinement and connectivity stage is used to generate smooth lines along the patterns of the segmented masks.

- We compare our framework's performance with several semantic segmentation methods using our dataset to demonstrate our superior performance.
- We create a research dataset for land surveying of painted traffic lines. The dataset is suitable for deep learning feature semantic segmentation and line fitting studies.

1.3 Thesis Outline

The rest of the thesis further motivates and describes our developed algorithms, datasets, and validations results, and is organized as follows. Chapter 2 discusses background research and related works, setting the foundation for our study across five distinct research scenarios. Chapter 3 presents our published algorithm for texture-less object detection and 6D pose estimation using natural images and depth maps data. Chapter 4 describes our proposed video traffic analysis system for local-area, in-situ contexts and its adaptability for new scenes. Chapters 5 and 6 demonstrate our published method for automatic land survey using multiscale voxelization and learning hierarchical features on LiDAR. Chapter 7 describes how to extract linear patterns from UAV imagery for road surveys. Finally, Chapter 8 summarizes our research findings and directions for future research.

Chapter 2

Background and Related Work

2.1 Object visual perception, positioning and pose

In this chapter, we embark on a review of the literature, delving into various approaches of computer vision and image analysis. Simultaneous recognition of an object and the estimation of its 6D pose have garnered significant attention in recent years. Object recognition involves categorizing real-world objects within a scene by extracting distinct features from the Region of Interest (RoI) and using these features to classify the object's category through a pre-trained model. Pose estimation, on the other hand, focuses on determining the position and orientation (pose) of a known 3D object in relation to the camera within the scene. Typically, a rigid object pose is represented by a 6 Degree of Freedom (6DOF) transformation matrix, comprising three translation and three rotation parameters, thus earning it the name "6D pose estimation." Existing techniques can be categorized as feature/template-based and learning-based approaches. The traditional approach for 3D object recognition is through template matching. Although the time performance of template matching is not suitable for real-time applications, it does not need a lot of samples and can recognize new objects by comparing them with a database [2]. The methods proposed in [3] and [2] are based on efficient template matching. They used optical images to detect objects. The authors in [2] also showed that occlusion could be less problematic by adding depth information. Their feature set contained surface normal gradients and orientations of the contours. These techniques worked best for texture-less objects of heterogeneous shapes. However, as we have pointed out before, industrial parts are usually symmetrical and have simple homogeneous shapes. Region-based approaches are used extensively in object recognition and pose estimation for 2D images. The author in [4] used active contouring to segment an image into background and foreground. They estimated the object's pose by using multiple

local appearance models. These models can capture spatial variations and therefore work well for heterogeneous objects. In [5], the authors proposed a framework to estimate the 6D pose of an object from a single RGB image. Their method iteratively reduced the uncertainty in object coordinates and object prediction. In order to deal with missing depth information, they computed an approximation of the 3D object coordinate distribution over depth maps. Another region-based approach proposed in [6] estimated the pose of an object using the local color histogram of a single RGB image. This approach fails to produce reliable results if the objects do not have color or textural information. More recently, Artificial Neural Networks (ANN) have gained popularity in solving computer vision problems [7]. Current approaches for object detection are based on bounding boxes, reconstruction of features in each detected bounding box, and high accuracy classifiers. These approaches are driven by the success of "region proposal" methods and region-based convolutional neural networks (RCNNs) [8]. Fast R-CNN [9] and Region Proposal Network (RPN) + Fast R-CNN [10] are more effective and accurate than the original region-based CNNs. Despite their strengths, these methods are computationally expensive. Recent approaches use pre-trained networks that accelerate convergence, achieving noticeably better time performance during object detection. These methods are often built upon the SOTA architectures for 2D object detection, such as Inception or ResNet [11, 12]. Using robust baseline systems, Fully Convolutional Network (FCN) has improved the outcome of object detection and semantic segmentation. These methods are conceptually intuitive and offer flexibility and robustness. They also provide fast training and inference time. A novel method, Mask R-CNN [13], extends Faster R-CNN by adding a branch for predicting segmentation masks on each RoI, parallel to the existing branch for classification and bounding box regression. The mask branch is a small FCN applied to each RoI, that predicts a segmentation mask in a pixel-to-pixel manner. Mask R-CNN is simple to implement and train, by extending the Faster R-CNN platform, which facilitates a wide range of flexible architecture designs. Although the mask branch offers a fast system and practical experimentation, a major limitation of these deep learning-based approaches is the need for a large volume of representative training samples for highly accurate performance [14]. In [1], we introduced an automatic approach for recognizing texture-less objects in industrial applications. In this approach, after localizing the objects in the 2D images, we extracted HOG features to train an object recognition algorithm. The object pose was obtained based on a point cloud matching process that used Fast Point Feature Histogram (FPFH). Despite our accurate results, the performance time of FPFH can be affected significantly by the number of points. A high number of points involves more operations to compute the

features, while a low number of points might not retain enough information, thereby affecting the final pose. We introduce several improvements on our latest approach [1]. More recently, there have been number of approaches using Artificial Neural Networks (ANN) for 3D object recognition and 6D pose estimation. In [15], the authors used RGB-D features to train a Convolutional Neural Network (CNN) model. The depth features added information about the horizontal disparity, height above the ground, and angle with gravity for each pixel. Data augmentation is a useful process for learning-based models to create a comprehensive and rich training set. For instance, authors in [16] augmented the data to create a training set for autonomous navigation. But, even with training samples, the adjustment for 6D pose estimation applications can be very time-consuming. [17] proposed a robust and scalable object detection technique that combines CNN with a region-based proposal to localize and detect objects. Many researchers modified and used this network to recognize and detect objects for autonomous driving and object localization [18–22]. However, almost all of these networks rely heavily on the textural information of the objects. The network proposed by [22] can handle irregular-shaped, texture-less objects. But this is not suitable for industrial parts where different parts can be very similar in shape. [21] proposed that a CNN regression framework is more suitable than a CNN classifier framework for 6D pose estimation because the pose space is continuous and therefore the pose estimation problem is also a continuous problem in nature. [23] predicts the poses of objects using only RGB images. They first segmented the 2D images to localize the objects of interest and then used a CNN model to predict the 6D poses of the objects. However, since they did not use any depth information or depth cues, they could not accurately predict the full 6D pose. More recently, a Rao–Blackwellized particle filter for tracking 6-D object poses has been proposed [24] and achieved promising results; however, it can be affected by heavy occlusion and each object requires a single autoencoder.

2.2 Temporal object detection and tracking

Real-time object identification has emerged as a vital component in a wide range of video applications. Numerous approaches have been proposed for object detection and tracking in recent decades. Traditional object detection methods were based on descriptors. Several descriptors have been proposed by researchers. They can be categorized into two groups: region-based and contour-based descriptors, depending on pixel values or curvature information [25]. Shape matrix [26], Zernike moments [27], convex hull [28], moment-based descriptors [29], and media axis features [30]

are some region-based shape descriptors that have been proposed in the literature. Moment based descriptors are very popular. Studies have shown that they are usually concise, computationally cost effective, robust, and easy to compute, as well as invariant to scaling, rotation, and translation of the object. However, it is difficult to correlate higher order moments with the shape's salient features due to the global nature of these methods [27]. The media axis features are robust to noise but are computationally expensive due to their capability of reducing information redundancy. Some of the proposed contour based shape descriptors are harmonic shape representations [31], Fourier descriptors [32], Wavelet descriptors [33], chain code [34], curvature scale space (CSC) descriptors [35], spectral descriptors [36] and boundary moments [37]. Fourier and Wavelet descriptors are stable over noise in the spectral domain, while chain code is sensitive to noise. CSC descriptors capture the maximal contour known as the object's CSC contour. This approach is robust to noise and to changes in scale and orientation of objects but does not always produce results consistent with the human visual system [38, 39]. Visual recognition is another key factor used to improve the accuracy of object detection and recognition. Techniques such as histogram of oriented gradient (HOG) [40], local ternary patterns (LTP) [41], local binary patterns (LBP) [42], scale invariant feature transform (SIFT) [43], binary robust independent elementary features (BRIEF) [44], speed-up robust features (SURF) [45], and oriented fast and rotated brief (ORB) [46] have been proposed over the years. However, these techniques still have insufficiencies. For example, various images of a specific object may appear profoundly different due to changes in the orientation of the object and the lighting conditions [47]. SIFT and ORB are robust descriptors that facilitate the object recognition. Although SIFT is slower than ORB, it is more stable. Both SIFT and ORB are suitable for real-time applications. However, these methods rely on textural patterns, and if the objects do not have enough textural information, like most of the industrial robotic automation and machine shop applications, they will fail to detect objects accurately.

In order to improve the accuracy of traditional object detection methods, researchers later combined machine learning techniques with shape descriptors [17, 48]. Many machine learning algorithms, such as support vector machine (SVM) [49], decision trees (DT) [50], linear discriminant analysis (LDA) [51], Naive Bayes (NB) [52], random forest (RF) [53], learning vector quantization (LVQ) [54], k-means [55] and k-medians [56] have been tested and have shown their efficiencies in solving classification problems.

Other object recognition approaches like LINE2D [2] and its variants, examine a template of sparse points across a gradient map. These approaches capture the target

shape directly without extracting the edges. The trade-off is a high false positive rate due to the lack of edge connectivity information in the object recognition process.

The YOLO (You Only Look Once) framework has distinguished itself for its exceptional balance of speed and precision. YOLO architecture has gone through multiple iterations, each improving on the strengths of its predecessors. In 2016, YOLOv1[59] was the first to deliver real-time object detection by predicting classes and bounding boxes in a single pass. In 2017, YOLOv2 (YOLO9000)[58] and YOLOv3[60] enhanced accuracy with refined designs, anchor boxes, and multi-scale detection. CSPDarknet53 and PANet were introduced in YOLOv4[61] (2020), while PyTorch integration and AutoAnchor optimizations were included in YOLOv5[62]. YOLOv6[63] by Meituan Vision AI Department continued the series' advancement in 2022. It included a PAN neck, an efficient backbone with RepVGG or CSPStackRep blocks, and a hybrid-channel method in the head. Better quantization approaches increased speed and accuracy, outperforming previous versions and competitive models. YOLOv8[64], introduced in early 2023, improved the series with scalable versions, enhanced feature fusion via a Coarse-To-Fine (C2F) module, an anchor-free model with a decoupled head for accuracy, and a YOLOv8-Seg for semantic segmentation.

Detection Transformers (DETR)[65], introduced in 2020 by researchers at Facebook AI Research (FAIR), revolutionized object detection by exploiting transformer designs, eliminating anchor boxes, and enabling end-to-end training. Set prediction loss was incorporated in the original DETR for comprehensive learning. Deformable DETR and Context Transformer variants improved efficiency and accuracy, while Deformable DETR+ and Progressive Transformer refined spatial modelling and object handling. Real-Time Detection Transformer (RT-DETR)[66] addressed computational issues by maintaining multi-scale features effectively, introducing IoU-aware query selection, and allowing for adjustable speed modifications. This evolution exemplifies DETR and RT-DETR's transformative impact in making object detection accurate and efficient.

On the other hand, robust and efficient multi-object tracking algorithms are the pillars of a good traffic analysis system. Their significance resides in their capacity to associate vehicle instances across time, allowing for traffic dynamics such as congestion patterns, vehicle counts, and near-miss occurrences to be extracted from video data. Most tracking algorithms are detection-based, where objects of interest are first detected within the frame using an object detection algorithm like YOLO or SSD, and then object track initialization and association are done using those detection hypotheses. Several approaches use a Kalman filter [67] estimator for object tracking. The Kalman filter is efficient in estimating the state of a dynamic system based

on noisy measurements over time, and it is known for its optimal performance under certain assumptions. Intersection over Union (IoU) tracker algorithm [68] uses the detection hypothesis from the previous and current frames, and associates vehicle tracks with detections based on IoU calculations. In scenarios where the system dynamics are highly nonlinear or the noise characteristics are not well-modeled by a Gaussian distribution, researchers adopted alternative filtering techniques such as the Unscented Kalman Filter (UKF)[69] or Particle Filter [70]. These filters are designed to handle nonlinearities and non-Gaussian noise better than the standard Kalman filter. Traditional algorithms, such as Mean Shift[71], KLT Tracker (Kanade-Lucas-Tomasi Tracker)[72], and TLD (Tracking-Learning-Detection)[73], have long been the foundation of tracking systems, relying on handcrafted features and heuristic methods. Mean Shift algorithm tracks an object by finding the mode of the color distribution in the target region. Camshift[74], an extension of Mean Shift, enhanced performance by adapting the window size during tracking. KLT Tracker consists of a feature-based tracker that tracks keypoints in the image. MOSSE[75] is a correlation filter-based tracker that uses a pre-learned filter for tracking. In contrast, modern algorithms leverage the power of deep learning, incorporating neural networks and sophisticated architectures to enhance tracking accuracy and robustness. Examples of modern algorithms include SORT, DeepSORT, YOLO-based trackers, and MOTNeRF. SORT (Simple, Online, and Real-Time) [76] uses a Kalman Filter [77] and the Hungarian Algorithm [78] for future state prediction and object association. Researchers have combined neural nets and traditional feature extraction methods to achieve SOTA results. The researchers behind DeepSORT [79] pioneered the inclusion of deep learning for appearance embeddings. They replaced the association metric in SORT with one that combined both motion and appearance information, improving robustness against misses and occlusion. StrongSORT [80] updated DeepSORT with more recent deep learning techniques in object detection and appearance embedding; they also added an appearance-free linking algorithm to address missing associations, and Gaussian-smoothed interpolation to address missing detections, ultimately achieving SOTA results. Other modern trackers are MOTSA (Multiple Object Tracking with Simultaneous Association)[81], Fantrack[82], CenterNet[83], GOTURN (Generic Object Tracking Using Regression Networks)[84], and SiamRPN (Tracking Objects without Labeled Data) [85]. These algorithms vary in terms of their complexity, performance under different scenarios, and computational requirements. The choice of an algorithm depends on the specific requirements of the tracking task and the available computational resources.

2.3 Point cloud object localization

Moving on to the challenges of object detection and classification in wide area monitoring, noticeable research efforts led to the rapid evolution of successful object detection algorithms for images. This success is also attributed to the promises of CNNs and the development of graphics processing units (GPUs). Initial approaches for object classification relied on hand-crafted features to recognize a pattern in an image or point-cloud. The latest approaches are based on deep learning algorithms with noticeable advantages over hand-crafted features methods, provided that a sufficiently large volume of representative training data is available. Unlike hand-crafted features that are defined on the basis of application requirements and domain knowledge, deep learning distinguishes itself by the ability of automatic feature learning from the training data. Nevertheless, due to the irregular nature and unordered data format of point-cloud, image classification algorithms, cannot directly take point-cloud data as input. As a result, one approach is to convert 3D point-clouds to regular 2D image representations and then apply a CNN to the converted or rasterized point-cloud. This is categorized as a multi-view-based method. MVCNN [86] is a pioneering work, which performs max-pooling multi-view to synthesize features from the views into a global shape descriptor. However, max-pooling cannot retain smaller feature elements from a specific view, resulting in information loss. Some researchers have proposed the GVCNN architecture [87], that better discriminates among views by using a complex view-pooling module (with view grouping and fusion) instead of the max view-pooling layer to more effectively aggregate views. The author in [88] proposed a Multi-view harmonized bilinear network (MHBN), which integrates local convolutional features by using harmonized bilinear pooling and a polynomial kernel to produce a compact global representation. In addition, several other methods have also been proposed to improve recognition accuracy. Some methods voxelize the point-cloud into 3D grids and then apply a 3D CNN. These are called "volumetric-based methods." The authors of Voxnet [89] proposed a volumetric occupancy architecture for 3D object classification. This method demonstrates as quite robust to the different representations. Also, 3D ShapeNets [90] learn the distribution of the point sets from various 3D shapes. However, the memory and computational cost of these methods are adversely affected by the high resolution of point-clouds. More recently, researchers introduced methods that directly work on raw point-clouds without any voxelization or projection to 2D images. These point-based methods do not introduce explicit information loss and have become increasingly popular. A pioneer in this category is PointNet [91], which directly takes a point-cloud as input. Then, a multi-layer perceptron (MLP) learns

point-wise features and finally extracts global features with a final max-pooling layer. The features are learned on each point, which causes information loss between points. Later, PointNet++ [92] overcomes this problem by using a sampling layer, a grouping layer, and the PointNet-based learning layer. This setup can capture the geometric features in the neighborhood of each point.

2.4 UAV imagery linear features extraction

The evolution and noticeable progress in remote sensing technologies has led many researchers develop algorithms for efficient semantic segmentation. This success is enabled by Deep Learning (DL), massive sensoring data, and the development of high computational performance systems with GPUs. DL algorithms, in particular, have been applied with outstanding results in object classification, semantic segmentation, object detection, and a variety of machine vision tasks. Various architectures represent promising avenues for the automatic extraction of street-line features painted on roads in aerial images, e.g., the center and edge-lines on highways. However, the extraction of painted lines can fail because of the complex backgrounds of aerial images, such as the impact of moving objects, animals, and occlusion caused by trees, power lines, cloud shadows, air pollution, and static or dynamic objects. Additionally, many land areas, e.g., bare ground, parking lots, rooftops, parcels, and rivers, typically share similar textures and structures as markings painted on the road, which makes them difficult to discriminate in many scenarios. All these factors represent a challenge for the automatic extraction of pavement painted lines in aerial images. Recent methods are widely available for road segmentation. Ma *et al.*[93] applied support vector machines (SVMs) to classify roads in high-resolution satellite images and extract their shapes from the binary representation using a mathematical morphology stage. Bond *et al.*[94] proposed a multistage simple color (luminance, saturation, and hue) space segmentation and Laplacian edge detection pipeline to extract roads automatically from satellite-taken images. For center-line and edge-line estimation, researchers have applied transform-based line detector [95], perceptual grouping theory with global optimization [96], and shape features and multivariate adaptive regression splines [97]. Yuan *et al.*[98] combined spectral and texture features using local spectral histograms with subspace projection for feature reduction. Similarly, Chaudhuri *et al.*[99] exploited both the spectral and spatial properties of roads using a multi-step approach which includes enhancement, segmentation, hole filling, small region filtering, noise removal method, and a linking stage. Liu *et al.*[100] proposed a road extraction method based on the shear transform, directional segmentation, road

probability, shape features, and a skeletonization stage. Anil *et al.*[101] introduced a statistical region-merging method for road segmentation, followed by a skeleton pruning by a discrete curve evolution process to extract road networks. Center-line extraction algorithms have been proposed in the past using SAR images. Cheng *et al.*[102] proposed a semi-automatic road center-line extraction method based on circular template matching. However, this method is not suitable for complex scenarios. Cheng *et al.*[103] later proposed a two-stage tracking method for extracting center points and center-lines iteratively. The algorithm consists of a local detection and global tracing strategy. In the study by Saati *et al.*[104], hand-crafted features were obtained for potential road area extraction through fuzzy inference. After that, a morphology skeletonization operation formed the road center-lines, and interest seed points were connected. Similarly, Cheng *et al.*[105] extracted road networks using a valley-finding method with multiple parallel particle filters and seed points. Other works [106–108] proposed morphological feature- based methods to find paths in urban roads. While these methods had acceptable performance, they usually suffered from a lack of robustness to occlusions, light, and contrast variations. Handcrafted feature-based methods suffer from the challenges of occlusions, limited adaptability for different data sources, and exhaustive data preparation and parameter selections.

More recently, algorithms based on CNNs have become popular in semantic segmentation of high-resolution aerial images. The current approaches achieve road segmentation using CNN encoder-decoder structured models, which are able to capture large spatial context, mainly for remote sensing applications. A variety of networks for semantic segmentation [109–111] exist, which have been widely adopted for line features extraction. These architectures have also shown acceptable performance in different applications such as autonomous vehicles for snow environments, ship identification, and water body extraction [112–114]. Road and center-lines are active research topics due to the high demand on remote sensing applications. Cheng *et al.*[115] proposed a cascaded CNN (CasNet) method to simultaneously cope with roads and determine center-lines; finally, a thinning algorithm was applied to obtain smooth, complete, and single-pixel width lines. Zhong *et al.*proposed a pipeline using FCN for road extraction from high spatial resolution imagery. The authors combined shallow fine-grained pooling layer outputs with a deep final-score layer to improve accuracy. Ventura *et al.*[116] proposed a CNN-based pipeline that predicted relationships between the central pixel and border pixels of an input patch. Then, they gathered the global topology of the road by iterating around the local connectivity. More recent methods combined the advantages of residual learning and skip connections to improve accuracy and performance. Bastani *et al.*proposed a CNN-based iterative

search to construct road network graphs. Zhang *et al.*[117] introduced a method with multi-scale features and enlarged the receptive field in the residual blocks for the road segmentation task. Lu *et al.*[118] proposed a globally aware network, CoANet, that learnt the segmentation and pair-wise dependencies to improve the road extraction performance. CoANet alleviates the occlusion problem with a connectivity attention module (CoA) to explore the relationship between neighboring pixels. Lu *et al.*[119] proposed a global-local adversarial learning pipeline for cross-domain road detection in high resolution satellite imagery. Similarly, Zhu *et al.*presented a framework for road extraction using a Global Context-aware and Batch-independent network. Lu *et al.*[120] exploited symbiotic relationships between the road and traffic lines to boost the road connections and enhance road completeness. Although previous methods can automatically learn contextual features, they only focus on extracting roads and not specifically painted traffic lanes. Some more recent methods used road center-lines to establish relationships with highways, benefiting the completion of the resulting road networks. However, as in our case study, these methods do not automatically extract classes like the center-line and road edge-lines. Most existing algorithms focus on high-resolution satellite images, which tend to lose the attributes of traffic lines on the roads due to lower resolution. Survey Automation is an emerging topic because it is required not only to segment patterns of an image, but also to map these pixel-level predictions to standard forms such as 2D/3D lines with geospatial information. The resulting lines must compare with the existing ground truth data (2D and 3D design data and metadata) for the same scenes previously extracted with conventional techniques. Since the existing databases ignore the lines painted on the roads, we created our database to evaluate the proposed method. In this thesis, we propose a system based on CNNs and a cascade of filters to refine and connect the traffic lines painted on long-distance rural roads in drone images.

Chapter 3

Object detection and 6D pose estimation in-situ

3.1 Introduction

Object localization and pose estimation are challenging but often essential processes in the field of computer vision. Many applications, such as robotic controlled pick-and-place, virtual reality (VR), augmented reality (AR), and autonomous driving, depend heavily on the accuracy of object recognition and pose estimation. These applications require a robust, accurate, fast, and efficient system that can handle a dynamic scene and learn new object instances. The system should also be adaptable and scalable to different environmental settings. In general, object recognition refers to classifying multiple instances of real-world objects in an image. Object recognition techniques rely on extracting a set of discriminative features from the Region of Interest (RoI) and feeding them into a previously trained model or computational formulation to recognize the object's class. On the other hand, pose estimation algorithms determine the position of a target object within a scene with respect to either the camera perspective or a designated reference point. The Six Degree of Freedom (6DoF) transformation matrix describes the pose of a rigid object, which is composed of three translation and three rotation parameters. Hence, it is called "6D pose estimation." Although object detection has been studied extensively for textured objects, where the objects can be represented by a sparse set of discriminative material pattern and color features , it is still an open research topic for industrial objects, which are texture-less, often symmetric, and made from the same material. There is an increasing demand for accurate detection of these types of objects, which are commonly found in industrial manufacturing and production processes, e.g., a robotic vision system requires pick-and-place, parts navigation, and object inspection

routines. Knowing the pose of the target object (e.g., spark plugs, nuts, or gears) can facilitate the end effector to precisely pick up the object. Texture-less objects can only be described by their global shape features, such as edges and depth cues [2, 3, 121]. The detection methods based on photometric local color and material patterns fail for these objects [17, 18]. [122] extends the method of [2], using color information into the dominant orientation templates (DOT) to avoid false detections. This method outperforms the original DOT combining the color and shape matching scores with logistic regression and improves the runtime. However, a limitation is that it cannot distinguish between colors with the same hues but different saturations, failing in scenarios where multiple objects have similar colors.

Some previous works used homography to determine the pose of an object. Note that these methods work only on piece-wise planar objects, which are common for image rectification, image registration, or computation of camera motion. However, when a variety of objects are close together, occluded, or lacking texture, the homography approach does not deliver satisfactory results. Other researchers proposed object detection and pose estimation methods using both RGB and depth map data [2, 3, 15, 16]. Although these methods usually outperform the optical-based approaches, depth sensors have a limited capturing range and are more sensitive to illumination conditions and reflective surfaces. They are more suitable for indoor environments. As a result, other researchers estimated a 6D pose from 2D images [5, 23, 123, 124]. Many of their works did not use depth sensors for outdoor environments, as they believed that depth sensors can be affected by the changing lighting conditions. However, even optical data can be affected by changing lighting, as well as by the presence of noise. Our system is based primarily on RGB-D data because most current robotic systems already have this capturing capability in place. Besides, most commercial and modern RGB-D sensors can achieve a range from 0.2 to 20 m in depth, matching the requirement of commercial robotic arms, which normally reach less than 20 m in length. Given these scenarios, our system achieves more accurate and reliable outcomes in detecting texture-less objects. We take advantage of the depth maps to segment the objects and use the RGB images to determine the optimal poses. Experimental results demonstrate that our approach is efficient and suitable for recognizing texture-less industrial objects. Our processing pipeline, shown in Fig. 3.1, consists of the following major components: RGB-D image segmentation and localization, feature analysis, and object class prediction and pose estimation.

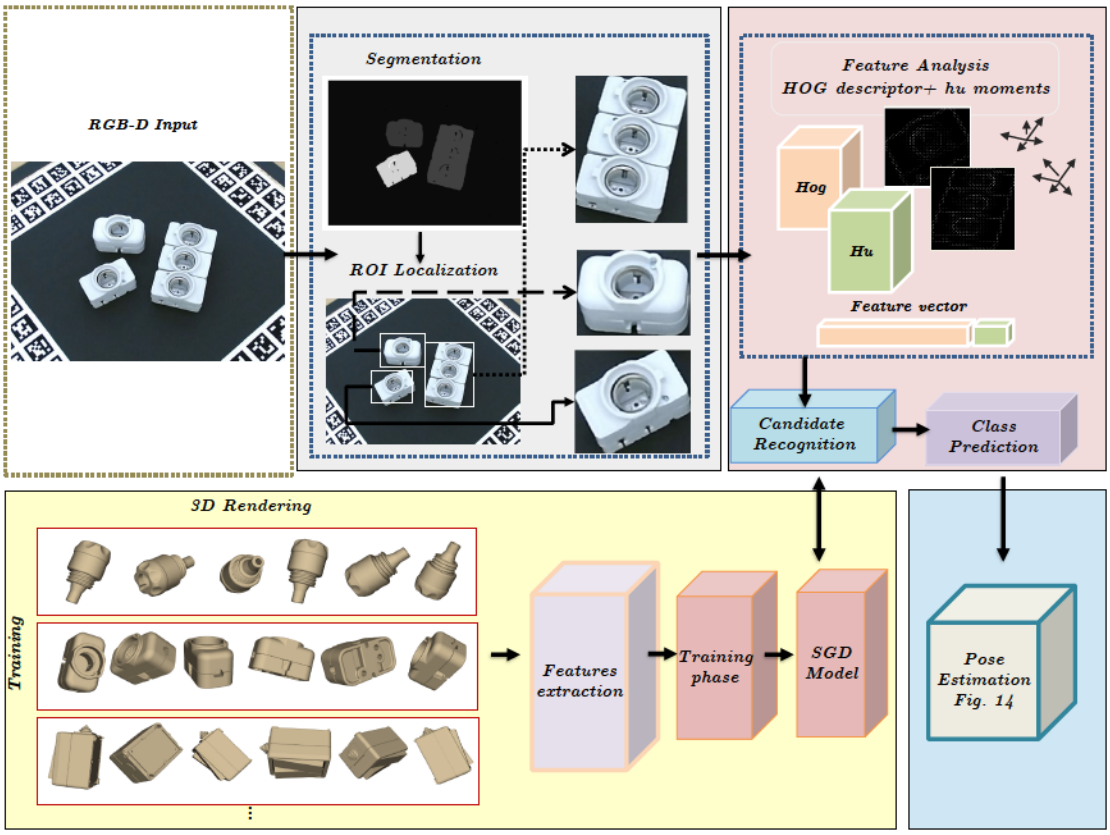
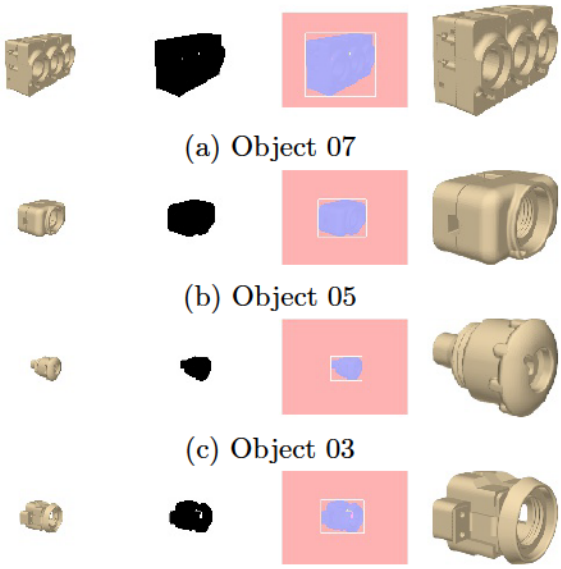


Figure 3.1: Our proposed object recognition processing pipeline. The RoI from the RGB-D input image is segmented, and then the minimum bounding box of each object is generated for feature extraction and object class prediction (top). Images of each object's CAD model are generated in x,y, and z axes for training purposes (bottom).

3.1.1 Texture-less object datasets

Different datasets for 3D object recognition and pose estimation have been introduced in recent years. Hodan *et al.* [125] introduced the public T-LESS dataset for estimating the 6D pose of texture-less rigid industrial parts. The dataset contains 30 objects with similarities in shape and size. Some of the objects are parts of other objects in the dataset. Many authors have shown that recognizing these objects is challenging. Even the SOTA techniques may fail in different scenarios. We focus mainly on this dataset as it fits our application for recognizing texture-less industrial parts.



(a) Object 07

(b) Object 05

(c) Object 03

(d) Object 12

Figure 3.2: First column: rendered images of different input objects from the T-LESS dataset. Second column: Watershed segmentation technique applied on rendered images. Third column: regions of interest are detected in each segmented region. Last column: cropped region of interest. The output images are used for training to recognize objects and estimate poses at a later stage.

3.2 Proposed method

We introduce a new approach to address the problem of object detection and 6D pose estimation for texture-less objects. A major issue with texture-less objects is the lack of many visual identifications, such as color and texture, which increases the difficulty of the problem. Furthermore, industrial objects are often symmetric, causing perspective ambiguity. Therefore, we propose an approach combining RGB-D data and a number of robust techniques, which contributes to a more effective refinement of the final target object pose. Our method is trained using rendered images from different perspectives of each CAD model. We first detect the 2D *RoI* in the input image using a clustering-based segmentation method. For the object recognition stage, we apply HOG algorithm with invariant moments to predict the object class in the input image. The pose estimation of the object is performed on each RoI_i, and we implement the DTW algorithm to find the optimal candidate pose of the object from a database. In this section, we explain the processing pipeline of our proposed approach and show the different stages involved to achieve our goal.

Manual labeling of different poses of an object is labor intensive and requires precise equipment to get accurate measurements. Training on rendered images of each

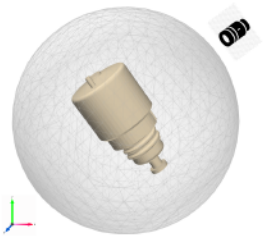


Figure 3.3: A spherical space view used to automatically render a 3D model of an arbitrary object from the T-LESS dataset. The object is rotated by an angle $s_d \in [0, 2\pi]$ along each of the x, y, and z axes with a step $s_d = 10$ degrees in our implementation.

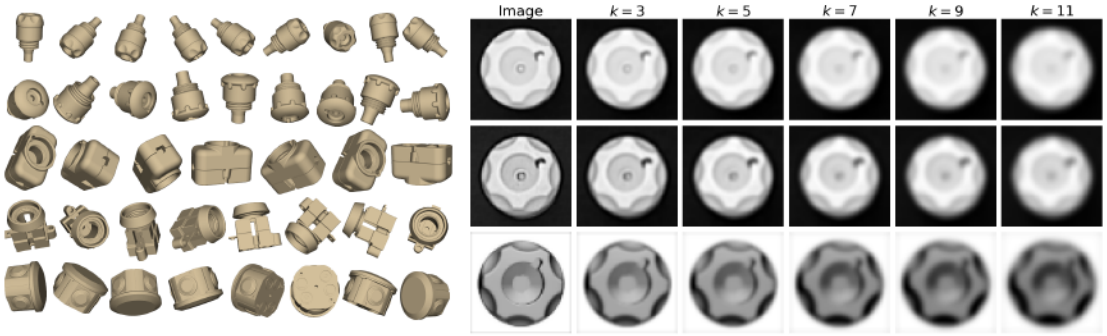


Figure 3.4: Left: Examples of the rendered images from five CAD 3D models of the T-LESS dataset. Right: Comparison of Gaussian blur filter with different kernel matrix size, i.e., 3x3, 5x5, 7x7, 9x9, and 11x11, for the same object. Top row: original image from the PrimeSense sensor and the resulting images with different kernel sizes. Middle row: images from the Kinect sensor. Bottom row represents our filtered images, which look more realistic.

CAD model object potentially benefits the pose estimation results. This process can generate a wide range of synthetic images, a confusion matrix of each view as a pose label, and object class label.

We first created a virtual scene with an empty white background and ambient light with uniform intensity and shadows. Then, we placed each CAD model at the center of the scene for further processing. The 3D model was rotated by a step $s_d \in [0, 2\pi]$ along the x, y, and z axes respectively, as shown in Fig. 3.3, with $s_d = 10$ degrees in our implementation. The distance of the object from the camera was kept constant at 400 mm. The output image size is 640×480 in color space. To obtain each rotation, we computed $\mathbf{R}_x\mathbf{R}_y\mathbf{R}_z \cdot (x, y, z)^T$. The Euler angle θ_x, θ_y, θ_z, and the corresponding rendered images were stored for future pose estimation. Examples of training views for different objects are shown in Fig. 3.4 (Left).

Gaussian filter to boost synthetic images

In this section, we briefly discuss how applying a Gaussian filter on the rendered images increases the similitude of the synthetic image to real image. In image processing, the Gaussian blur filter is commonly used to smooth a given image $\mathcal{I}$. Using the filtering as a pre-processing step improves the qualitative representation of the 2D image. Synthetic images generated from a 3D model are often semantically different from the real RGB images. For this reason, we transform the rendered image to another similar representation using the Gaussian filter. Theoretically, we can apply a Gaussian blur to $\mathcal{I}$ analogous to the convolution of a 2D image with a Gaussian function as shown in Fig. 3.4. Based on our experiments with different kernel sizes, $\kappa = 3$ generates the best result compared to bigger kernel sizes, which tend to blur the image.

In this context, 2D Gaussian is defined by the product of two such Gaussian functions as shown in (3.1):

$$G(x, y) = \frac{-1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.1}$$

where x represents the distance from the origin in the horizontal direction, y is the distance from the origin in the vertical direction, and σ is the standard deviation of the data. These functions average the center of the pixel (which is the center of a kernel matrix) to generate a smooth visual effect in the image. However, Gaussian blur uses the weight concept in averaging the pixels. These weights are assigned to all the pixels according to normal distribution. The equation of the Gaussian blur is defined in (3.2).

$$GB[I]_p = \sum_{\mathbf{q} \in S} G_\kappa \left\| \mathbf{p} - \mathbf{q} \right\| I_\mathbf{q} \tag{3.2}$$

where, κ is the size of kernel matrix. Typically, kernel matrices have the size of $(2r + 1) \times (2r + 1)$. The integer r is the radius of the filter and is a parameter that controls the size of the neighborhood that is being averaged. If $\kappa = 2r + 1$, the kernel G_κ is computed as shown in (3.3).

$$(G_{2r+1})_{i,j} = \frac{1}{4^{2r}} \binom{2r}{i+r} \binom{2r}{j+r} \tag{3.3}$$

Pixels near the center have the highest weights, and the weights decrease when moving toward the edges. It can be seen from Fig. 3.4 that after applying the Gaussian filter our image (Bottom row) is more realistic than the sensor captured synthetic ones (Rows 1 and 2).

Segmentation in the filtered images of 3D models

To prepare the training data, we apply the Watershed transformation technique [126] to segment the target object in the RGB image $\mathcal{I}_R$. Watershed is a morphological technique for image segmentation that groups the pixels in $\mathcal{I}_R$ based on their similarity of intensity. It uses gray tone values of $\mathcal{I}_R$, which is interpreted as a topographic surface. During the sequential "flooding" of the surface minima of the gray values, the technique partitions the gradient image into watershed lines and catchment basins. The result of the Watershed transform produces closed object contours at low computational cost compared to other more sophisticated vision-based segmentation techniques. After segmentation, edges features in each segmented region are extracted to uniquely characterize each object.

In practice, this traditional transform often leads to over-segmentation due to noise in the data. In order to deal with over-segmentation, we use the marker-based Watershed approach. The Watershed transform floods an image of elevation, starting from the markers. However, it is necessary to pre-process the optical image to determine the catchment basins of these markers. The marker regions are pixels that we can label unambiguously as either object or background; found at the two extreme parts of the gray value histogram. The identified markers are used in the watershed segmentation method. We apply the fast and simple morphology Sobel operator to compute the amplitude of the gradient in an optical image. The Sobel filter uses two 3 x 3 kernels: one for changes in the horizontal direction, and one for changes in the vertical direction. The two kernels are convolved with the original image to measure an approximation of the derivatives. Even if the markers in the background are not well distributed, the barriers in the elevation map are high enough for these markers to flood the entire background. After that, we remove the small holes with mathematical morphology. Using these regions, we apply the classical marker-based Watershed transform to detect the objects in the image. As shown in Fig. 3.2, this approach provides good quality object segmentation with a low computational cost.

After segmentation, features in each segmented region are extracted to uniquely characterize each object. Edges are basic attributes of texture-less objects which make them simple to distinguish with human visual perception. In Fig. 3.2, all objects have same color but different shapes, resulting in different edges. For example, object 7 has three holes on top, while object 5 and object 12 have only 1 hole. Object 5 has rounded edges while object 12 has sharp edges. These variations on edges are prominent features that can be used to distinguish between different objects and perspectives. For this reason, HOG features are extracted in each RoI after segmentation. The HOG

method provides acceptable performance under different scenarios and applications. The HOG method extracts the local histograms of orientations and magnitudes of edges in an image or pattern in order to represent the shape of the target. Section 3.2.2 describes more about HOG features for object recognition.

3.2.1 Real scene images: initial segmentation and refinement

In this section, we explain the steps to segment objects in real input images where the scene can be cluttered. Image segmentation represents a fundamental and important step in image processing to understand content composition. A segmentation algorithm helps to split an image into semantically separated partitions, or clusters, belonging to specific pixel distribution in an image. While segmenting an RGB image can provide useful semantics, we observed that including depth maps can achieve a higher detection accuracy of the regions of interest when dealing with partially occluded texture-less objects. Many CNN-based architectures can perform well for this specific task but require a large number of training images.

Since the depth images do not contain texture information, segmentation is challenging, especially for cluttered scenes. We need to explore characteristics other than texture. Depth maps contain points, each having a 2D location that is associated with a distance value from the sensor. On the depth maps, a drastic change represents an edge or border that can distinguish the surface using morphological operations. Thus, we used a Sobel operator to detect the edges in the image. Sobel is a 2D operator that measures the spatial gradient on the input image using two convolutional masks of size 3×3, representing the gradient's estimation in the x-axis and y-axis. Each mask passes through the depth image $\mathcal{D}$, performing operations on a rectangular region of pixels one at a time.

We performed the segmentation stage using the depth map image $\mathcal{D}$. We obtained an initial binary mask $\mathcal{M}_i$. The lines of high contrast represent the regions where drastic changes occurred on $\mathcal{D}$. However, $\mathcal{M}_i$ contained linear gaps surrounding the RoI. Thus, we applied a dilation operation to fill the holes in the binary mask; $\mathcal{M}_i \oplus \mathcal{M}_b = \left\{ z \mid (\hat{\mathcal{M}}_b)_z \cap \mathcal{M}_i \neq \varnothing \right\}$, The structuring element, $\mathcal{M}_b$, is a matrix that identifies the pixel in $\mathcal{M}_i$ and defines the neighborhood used in the processing of each pixel. $\hat{\mathcal{M}}_b$ is the reflection of the structuring element $\mathcal{M}_b$. It removed all minima which were not connected to the image border. A final mask, $\mathcal{M}_f$, and RoI_i, were obtained during the initial segmentation process, as shown in Fig. 3.6. To refine the results of the initial segmentation on depth images, we used K-means clustering, which is an iterative unsupervised learning approach. Given the initial RoI_i on image

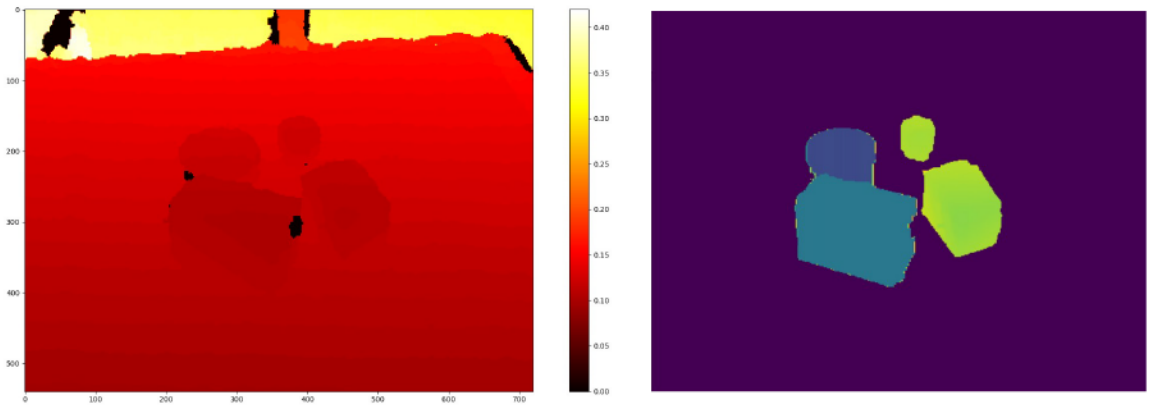


Figure 3.5: Left: Input depth image. Right: Resulted clusters computed from the depth map using our cluster-based segmentation step.

$\mathcal{D}$, we could find clusters by minimizing the sum of the square distance between the data and the correlated centroid of each cluster in the initialization step. In our case, the K-means algorithm classified a given set of depth map pixels into k sets of separate clusters. The K-means algorithm can distinguish the data in two steps. First, it measures k centroids in different locations of the image. Then, each point that is nearest to the cluster will be taken as part of it. Iteratively, the algorithm stops when no variation of the centroid occurs in the subsequent step, which indicates that the depth map points assigned to each cluster have become stable. These steps are described as follows: First, cluster k and its centroid C are initialized. Then, Euclidean distance $d = \|\mathcal{D}(x, y) - Ck\|$ is estimated for each depth map. The pixels closest to the center are assigned based on d. In every iteration, pixels are assigned to clusters and the new centroid for each cluster is recalculated. This process is repeated until the maximum number of iterations is reached, or every cluster center moves less than a tolerable error value. Finally, the clusters in the depth maps are projected onto a 2D image (Fig. 3.5). The K-means algorithm is very robust on depth maps, especially for scenes with cluttered backgrounds, because it does not rely on the color, texture, or pixel intensity but on the distance. It is also robust to indoor changing lighting conditions. Using this method, we do not rely on the learning approach for image segmentation like the CNNs. However, if we apply K-means on the whole depth map $\mathcal{D}$, post-processing is needed, because depth pixels that do not correspond to the target object can also be assigned to the cluster, generating a noisy region. Noise can be caused by external factors like the properties of the object and the physical environment. We therefore use a morphological technique to decompose the whole $\mathcal{D}$ image into multiple $RoIs$, which can remove some noise. Then, K-means is applied as a subsequent segmentation process to improve the detection of occluded objects.

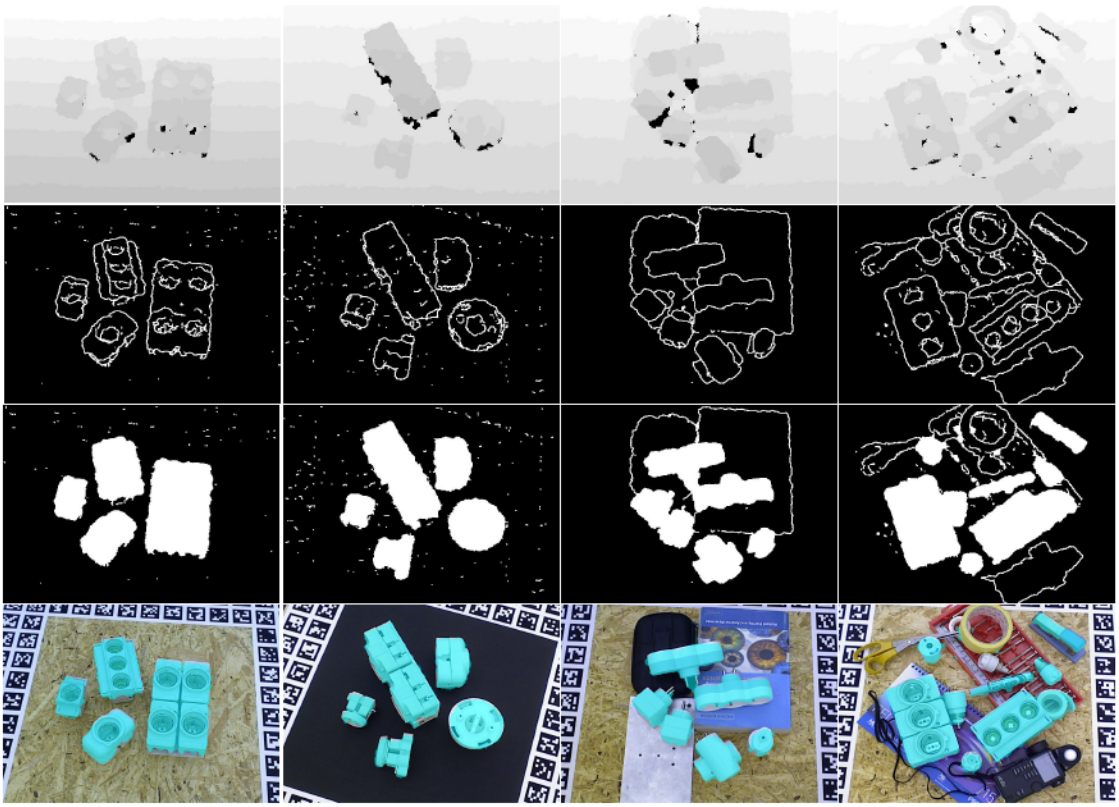


Figure 3.6: Comparisons of scenes ID 3, 11, 14, and 17 in the object segmentation process. Morphological operations are applied on the depth image to find RoI_i. First row: the input depth image of each scene containing multiple objects. Second row: the initial binary mask $\mathcal{M}_i$ after applying the Sobel filter in the x-axis and y-axis, respectively. Third row: the final mask obtained with multiple RoI_i. Bottom row: the original color image overlapped with the final mask $\mathcal{M}_f$ on each scene

Each detected cluster is processed as an individual patch, where the objective is to distinguish sub-regions inside every $RoI_i \in \mathcal{D}$.

3.2.2 Object detection

After segmentation, in the object recognition task, we consider grids of HOG as a feature descriptor. The technique behind HOG features was first proposed by McConnel [127]. This method has shown good results in applications that require object recognition, e.g., pedestrian detection. HOG provides a higher performance relative to other existing descriptors for the mentioned purpose. It measures the occurrences of the gradients in a dense grid that corresponds to a 2D image. Then, a histogram of gradients is generated. The dominant features of texture-less objects are edges or gradients. Dividing the image into a finite number of local cells (sub-

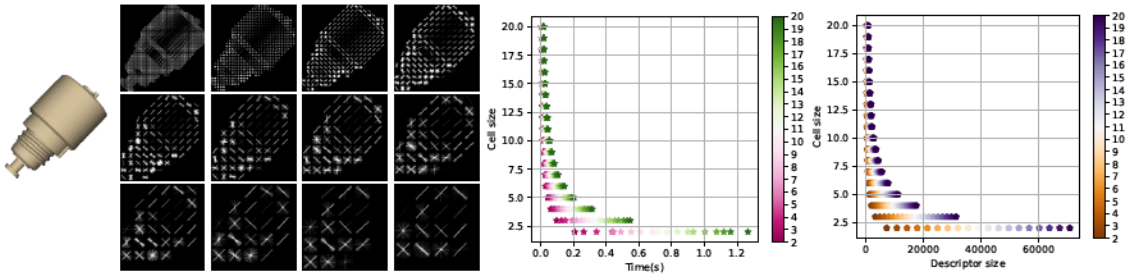


Figure 3.7: Left: Visualization results of HOG image on Object 4. Top row: resulting images with initial HOG parameters of cell size $s = [2 \times 2, 4 \times 4, 6 \times 6, 8 \times 8]$ and number of bins $b = [2, 4, 6, 8]$, respectively. Middle row: $s = [10 \times 10, 12 \times 12, 14 \times 14, 16 \times 16]$, $b = [10, 12, 14, 16]$. Bottom row: $s = [18 \times 18, 20 \times 20, 22 \times 22, 24 \times 24]$, $b = [18, 20, 22, 24]$. Right: Performance time (x-axis) vs HOG descriptor parameters: (cell size [2-20] (Left y-axis) and bins=[2-20] (Right y-axis)), applied on a 120*120px image sample (Object 4). Feature vector size (x-axis) vs HOG descriptor parameters: (cell size [2-20] (Left y-axis) and bins=[2-20] (Right y-axis)) applied on Object 4.

blocks) is a suitable approach to detect gradient directions and edge orientations for texture-less objects. The HOG method applies two fundamental filters, denoted by $w_x = [-1, 0, 1]$ and $w_y = [-1, 0, 1]^T$, on $\mathcal{I}$ to measure the horizontal and vertical gradients, which are $g_x = \mathcal{I} * w_x$ and $g_y = \mathcal{I} * w_y$, respectively. The magnitude of the gradient $\varrho(x, y)$ in the local region of $\mathcal{I}$ is computed and the corresponding gradient angle value at each pixel of I is estimated in (3.4).

$$\beta(x, y) = \tan^{-1}\left(\frac{g_y(x, y)}{g_x(x, y)}\right) \tag{3.4}$$

In each sub-block, the histogram feature stored at $H(\varrho, \beta)$ based on the gradient strength and the direction of every pixel is determined by the number of cells and blocks. We describe the function of these parameters as follows.

Cells: the input image is initially divided into rectangular regions of pixels sets c called cell size. Each cell has a fixed size defined by the initial parameters. For $\beta \in [0, 180]$, this range is divided into $\mathfrak{b}$ gradient orientation bins denoted by $\frac{\mathfrak{b} \times 180}{\mathfrak{b}_t}$, $\frac{\mathfrak{b}+1 \times 180}{\mathfrak{b}_t}$, where $\mathfrak{b} = 0, ..., \mathfrak{b}_t - 1$. The histogram of oriented gradient $\hbar$ results from determining cell-wise weighted voting. For each pixel, the ϱ and β are divided into two values based on the ratio to the two nearest bins. The same process is performed by adding ϱ for each bin that contains the corresponding β of the pixel intensity.

Block: Another important parameter is the block size. Each block contains a set of cells c. All blocks are overlapped and normalized c_{norm} in order to ensure tolerable contrast of normalization and robustness to illumination changes and noise. This process of histogram calculation is then repeated for each cell of the image. We apply

the HOG method on each rendered image in gray-scale space. The input image is the smallest rectangle containing the labeled region detected during the segmentation stage, denoted by the vector $[cx, cy, u, h]$, where cx and cy are the coordinates of the center of mass of the labelled region, and u and h are the width and height of the bounding box respectively. Then, the RoI_i contained in the bounding box is resized. We evaluate this method using different sizes, from 60×60 pixels to 150×150, to determine the best value for the HOG algorithm. An example of the evaluation for an image is shown in Fig. 3.7. This step reduces the computational cost of object recognition. In our experiment, we compared different cell size c_s of dimension $n \times n \in [2, 16]$ to find the best cell size that maintains a discriminative HOG (Fig. 3.7), while reducing the computational complexity. Based on the performance evaluation, we use the following settings for our final HOG detector: cell size 10 x 10 pixels, eight orientation bins, and one block normalization. Other features that we use are invariant moments, which have been a classical approach for object recognition. These invariant moments were first presented in the pattern recognition community by Hu [29], who deployed the results of the theory of algebraic invariants and derived the seven important invariant moments to object rotation in 2D space.

A two-dimensional $(p+q)th$ order moment is described in (3.5).

$$m_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} x^p y^q \mathcal{I}(x, y) dx dy \tag{3.5}$$

For corresponding image $\mathcal{I}(x, y)$ in RoI_i, the moments of all orders exist if the image is represented by a piecewise continuous bounded function. The moment sequence m_{pq} is uniquely determined by $\mathcal{I}(x, y)$; conversely, $\mathcal{I}(x, y)$ is also uniquely determined by the moment sequence m_{pq}. These features can be found using central moments, which are defined as follows:

$$\mu_{pq} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} (x - \bar{x})^p (y - \bar{y})^q \mathcal{I}(x, y) dx dy \tag{3.6}$$

where $\bar{x} = \frac{m_{10}}{m_{00}}$ and $\bar{y} = \frac{m_{01}}{m_{00}}$, represent the centroid of $\mathcal{I}(x, y)$. The centroid moment μ_{pq} is computed using the centroid of the image, which is equivalent to m_{pq}, whose center has been shifted to coincide with its centroid. The normalized central moments are defined as follows:

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}, \; \gamma = (p + q + 2)/2, \; p + q = 2, 3, ...,) \tag{3.7}$$

Based on normalized central moments, the seven invariant moments are:

$\phi_1 = \eta_{20} + \eta_{02}$

$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$

$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \mu_{03})^2$

$\phi_4 = (\eta_{30} + 3\eta_{12})^2 + (3\eta_{21} + \mu_{03})^2$

$\phi_5 = (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + 3(\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$

$\phi_6 = (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03})$

$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]$

These invariant moments have useful properties, including robustness under image scaling, translation, and rotation. The combination of HOG features and invariant moments forms the feature vector that we used for object classification. This combination allowed us to extract the inner and outer information of RoI_i. Hu moments capture the external shape features of RoI_i which partially help to overcome the problem of different perspective on object recognition. These invariant moments are unaffected by transformations. For example, assume that Object 4 in Fig. 3.7 rotates clockwise or counter-clockwise 10, 45, or 90 degrees on one axis, then for these transformations to correspond to very similar moments values. In another case, two objects of different classes may look similar, such as object 1 and object 2, both having similarities from a top perspective. In this scenario, Hu moments values for these two objects can also be similar, which might not help the training stage. That is why these moments are combined with HOG features; the HOG method describes the information within the shape of the object so that the final characteristic vectors for training are different between objects and perspectives. After calculating the Hu moments on target regions, these values are concatenated with the HOG feature vector (with parameters of cell size 10 x 10 pixels, 8 orientation bins, and block normalization) into a final composite feature vector for each RoI_i. These vectors are obtained off-line for later use during the training of model classification.

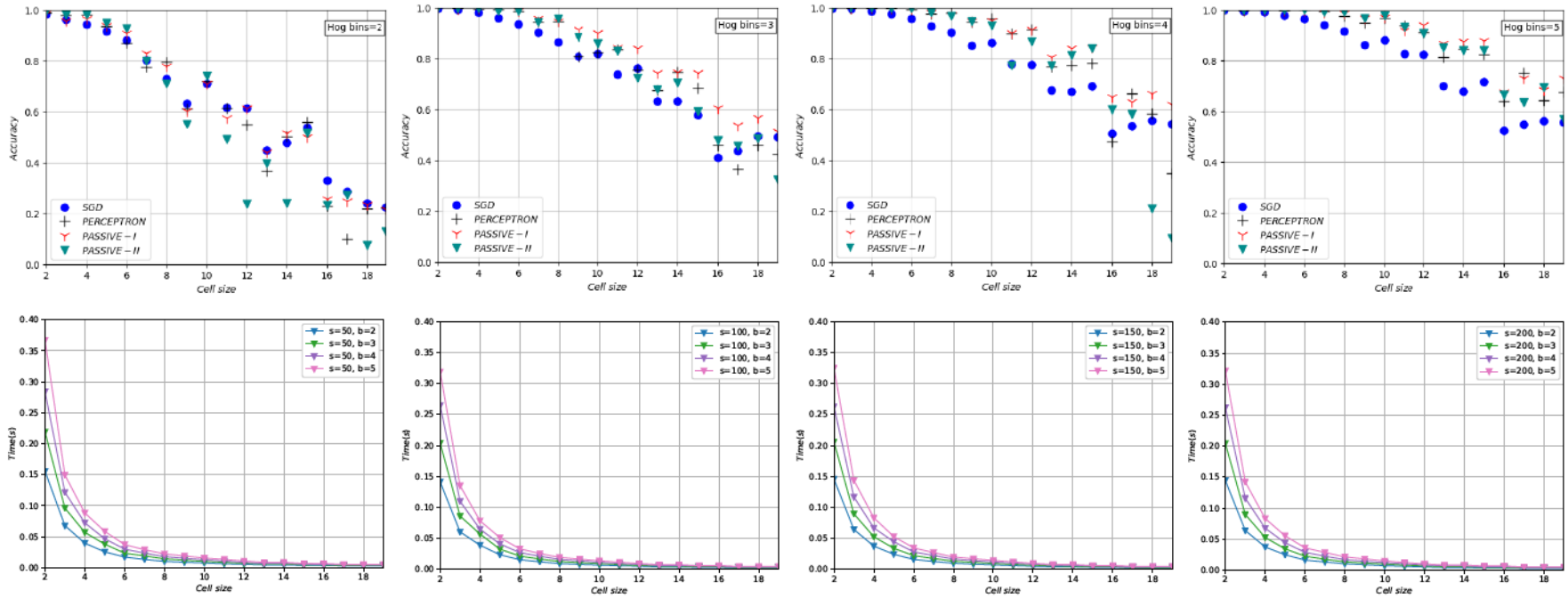


Figure 3.8: Comparing accuracies using different initial parameters of the Histogram of Oriented Gradients (HOG) method for training classifiers in object recognition. First column: shows the accuracy of different classifiers for object recognition by tuning HOG parameters. Second column: shows the relation between cell size $[2-18]$ and computational time (seconds) for different image scales $s = [50, 100, 200]$ and number of bins $b = [2-5]$.

3.2.3 Object class prediction

In this section, we discuss the training of the classification approach in our system. We used mini-batch gradient descent to train different models for classification. In our case, a feature vector containing HOG + invariant moments provides the samples for training. In mini-batch gradient descent, $T = \mathscr{P}/\mathscr{A}$ iterations are computed per training epoch. $\mathscr{A}$ represents the size of the mini-batch and $\mathscr{P}$ is the total number of training data. The weight parameters $\mathbf{v}$ are therefore obtained through optimization of the approximated values of the error function f, defined as:

$$E[f(\mathbf{v})] = \frac{1}{\mathscr{A}} \sum_{i=(t-1)\mathscr{A}+1}^{t\mathscr{A}} f(\mathbf{v}; x_i) \tag{3.8}$$

where $t \in 1, ..., T$ is the iteration index and x_i is the ith training sample. At each iteration the weights are adjusted using the gradient descent update rule $\mathbf{v}^{t+1} = \mathbf{v}^t - \varphi \nabla_{\mathbf{v}} E_t[f(\mathbf{v}^t)]$, with φ being the learning rate and $\nabla_{\mathbf{v}} E_t$ representing the gradient of the loss function. The output of these models gives a vector of probability estimate per RoI_i features, and the index with the maximum value corresponds to the class prediction. This class prediction is used to compute the pose estimation of the object.

Estimating the pose of the object is quite challenging because the camera can capture thousands of perspectives of the object. It becomes even more challenging when the object presents symmetries in some of its axes or when different objects show equal perspectives that do not allow differentiation between them.

3.2.4 3D pose estimation

Our goal is to estimate the 3D pose of an object by using the contour of its binary mask with respect to the camera. We assume that for each orientation of the object, a contour representing the pose of the object can be obtained starting from a reference point in the image, such as the centroid of the segmented region of the object. Determining the object's binary mask is important for estimating the closest pose more accurately. However, another essential factor that defines the final estimation is the matching algorithm.

We created a database consisting of the outlines of the binary masks of the images with different perspectives of each object, labeled with their respective degrees of orientation on the x, y, and z axes. We generated perspectives from 0 to 360 degrees around each axis with a variation step of 10 degrees instead of every degree between consecutive perspectives to reduce the total number of samples. This strategy covers a large part of the different orientations that an object can have in 3D space. We also

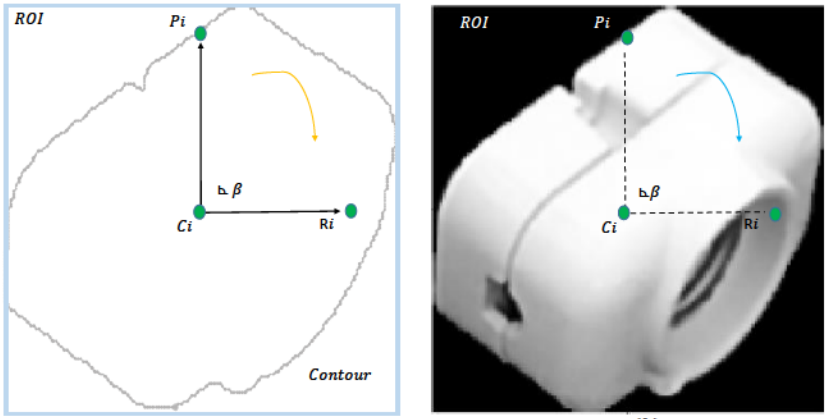


Figure 3.9: We determine the perpendicular line $CiPi$ to the line formed by points Ci and Ri. The intersection between the resulting line $CiPi$ and a point on the contour represents the initial index. All other points are numbered clockwise. Finally, the distance between the center Ci and each point of the contour is obtained for the matching process.

identified specific ranges for objects that show symmetries on the same axis, to avoid having equal contours for different rotational angles. The spatial similarity between two shapes, f1 and f2, can be defined as the minimum cost of feature matching from f1 to f2 or vice versa. The matching cost is related to the degree of dissimilarity in appearance, slope, corner, and spatial location. Normally, a shape-matching technique and distance measure, e.g., structural similarity index measure (SSIM), peak signal-to-noise ratio (PSNR), or iterative minimum mean square error (IMMSE), requires the computation of minimal cost correspondences between sets of features on two shapes.

Our objective is to detect the orientation of the object. Therefore, the technique must be variant to rotation, but invariant to scaling. For that goal, we implemented a modification of the marching cubes algorithm that is often used for displaying triangulated surface models from 3D medical data. We used the two-dimensional technique of this algorithm that approximates the structure of the industrial object by generating sets of voxels. Initially, squares are created to represent grids over the image. Then, the point of intersection of the straight lines with each other represents the structure at this specific location. This allows us to determine which points belong to the region. After performing this process for all the points, the algorithm's marching squares generate a boundary representation of a complex 2D structure, as shown in Fig. 3.9 (left). We assume that the starting point of the object's contour in

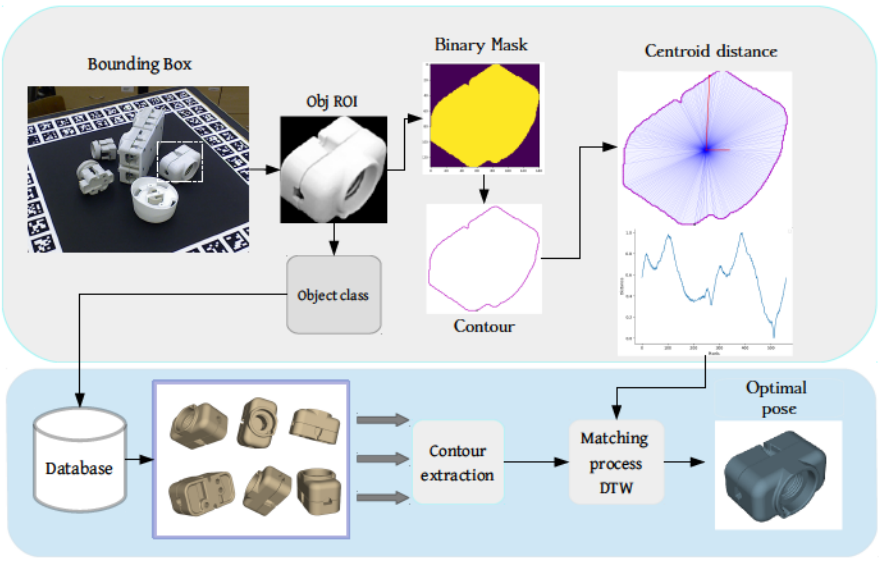
each of its orientations is located in a specific direction and position directly related to its bounding box and the extreme pixels of the segmented region. Our goal is to find matches in our database of the contours obtained from the rendered images. Using the centroid of the bounding box, as a reference, we proceed to determine the outline of the object. The starting pixel of the contour is automatically found at the top of the bounding box as shown in Fig. 3.9. The centroid and this starting pixel represent a line perpendicular to the horizontal axis of the bounding box. All other contour points are found clockwise until the region that delimits the binary mask object is closed. After obtaining all the contour points, these points are normalized to pass through the contour matching stage using a DTW algorithm to find the optimal matching contour, as shown in Fig. 3.10 and Fig. 3.11.

Finding similarities with DTW

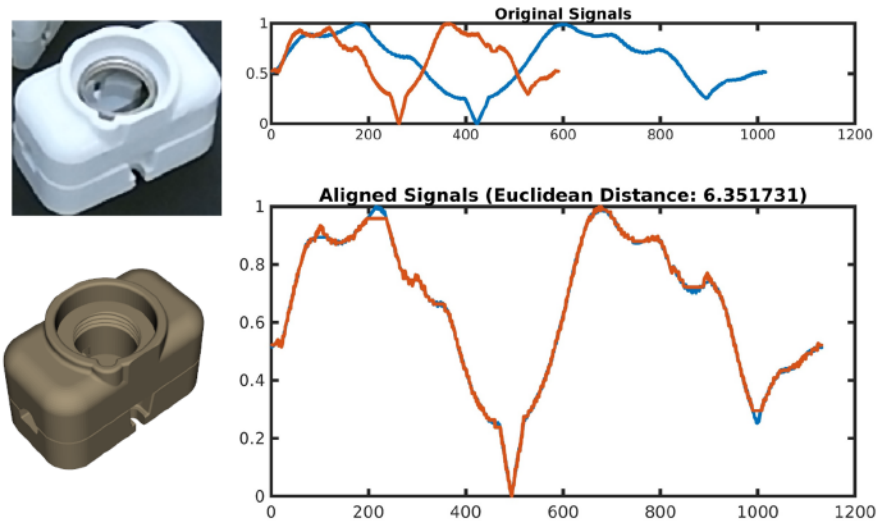
DTW is suitable for measuring similarities in temporal signals and classifying multivariate temporal sequences of data that have different phases or lengths over time. An example is comparing and aligning gait signals to predict falls using wearable sensor devices. The DTW algorithm has been adopted to various real-world applications such as speech recognition, handwriting, gait analysis, and data mining. Two given temporal signals with equivalent features arranged in the same order can appear very different, due to the durations of feature patterns. DTW reconciles these durations by comparing the corresponding features at the same location on a common time axis, which helps to highlight the similarities between the signals. We use the DTW algorithm to find the optimal match in the database and estimate the 3D pose of industrial objects. The time-series signal that passes through the DTW algorithm is the normalized difference between the centroid of the bounding box and the 2D location of each pixel of the external contour of the object. We compare the input denoted by $X := (x_1, x_2, \ldots, x_i, \ldots, x_N)$ of length $N \in \mathbb{N}$, and each candidate in our database denoted by $Y := (y_1, y_2, \ldots, y_i, \ldots, y_M)$ of length $M \in \mathbb{N}$. Despite the fact that X and Y can have different sizes, the DTW algorithm stretches both vectors to the same length. Then, it compares locally point-by-point with some warping within a small neighborhood along the time axis. The DTW distance is calculated as follows:

$$D(i,j) = dist(x_i, y_j) + min \begin{cases} D(i, j-1) \\ D(i-1, j-1) \\ D(i-1, j) \end{cases} \tag{3.9}$$

When applying DTW, one needs a local cost measure, which is also called the "local distance measure." It is used to compare the different features of the times series.



(a)



(b)

Figure 3.10: (a)Pipeline of our pose estimation process. The RGB-D input image is initially segmented into multiple *RoIs* in the localization stage. We use the bounding box of the object to predict the object class. Then, the contour of the binary mask is extracted to compute the distance between the centroid of the bounding box and each pixel along the contour. (b) Matching with our database objects is performed using the DTW algorithm to find an optimal match.

Typically, $c(x, y)$ is small (low cost) if x and y are similar to each other, and otherwise $c(x, y)$ is large (high cost). Evaluating a local cost measure (such as L_2, Euclidean

distance, or Manhattan distance) for each pair of elements of the sequences X and Y results in a cost matrix $C \in \mathbb{R}^{N \times M}$ defined by $C(n, m) := c(x_n, y_m)$. Then, the goal is to find an alignment between X and Y with a minimal cost as, shown in Fig. 3.10. We define the notion of an alignment as follows. An (N, M)-warping path (or, simply, a "warping path" if N and M are clear from the context) is a sequence $p = (p_1, ..., p_L)$ with $p_l = (n_l, m_l) \in [1 : N] \times [1 : M]$ for $l \in [1 : L]$ satisfying the following three conditions: (i) Boundary condition: $p_1 = (1, 1)$ and $p_L = (N, M)$, (ii) Monotonicity condition: $n_1 \leqslant n_2 \leqslant \cdots \leqslant n_L$ and $m_1 \leqslant m_2 \leqslant \cdots \leqslant m_L$, and (iii) Step size condition: $p_{l+1} - p_l \in \{(1, 0), (0, 1), (1, 1)\}$ for $l \in [1 : L-1]$. All matching elements i with a minimal cost lower than a threshold are considered candidate matches. Then, a structural similarity (SSIM) metric is applied to find the candidate with the best index, and finally retrieve the optimal pose in the database. The problem of template matching using the contour can be formulated as:

$$\dot{T} = \arg\min D(T_i, Q), T_i \in \tau \tag{3.10}$$

where $\tau = T_1, ..., T_N$ is a gallery of N template contours, and D is a distance metric. Our pose estimation strategy using DTW can still handle situations with partial occlusion, depending on the object. As long as the object of interest does not present occlusion in its upper part from the camera's perspective, our strategy remains effective, as shown in Fig. 3.15.

Optimal pose candidate

Once the initial matching stage has been completed, all selected candidates according to their DTW scores must go through another refinement process. The bounding box of each candidate pose is extracted. Both bounding boxes of the *RoI* in the input image and database are normalized and resized to 130×130 pixels. Both images are filtered by a linear differential operator, approximating the second derivative, which is also known as the "Laplacian operator." This filter discriminates the regions with intense changes, leaving only the edges of the images. Then, we compute the SSIM between the two images. SSIM compares the structural information of two images based on the luminance, contrast, and structural attributes of the images. The maximum score of 1 indicates that the two images are structurally the same, while a value of 0 indicates no structural similarity. The same process is repeated on each of the selected candidates. We take the highest similarity score to represent the optimal pose. The optimal pose computation pipeline is shown in Fig. 3.10.

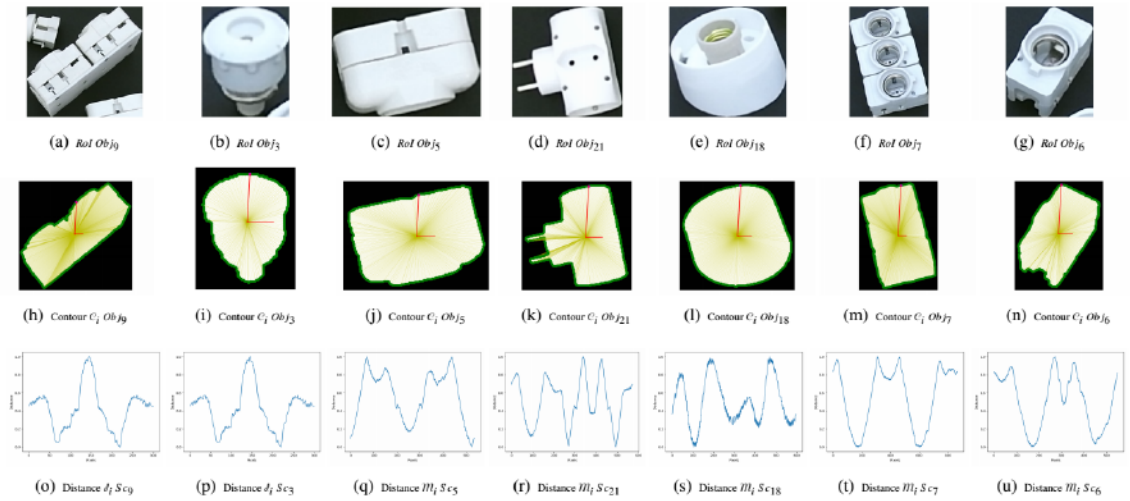
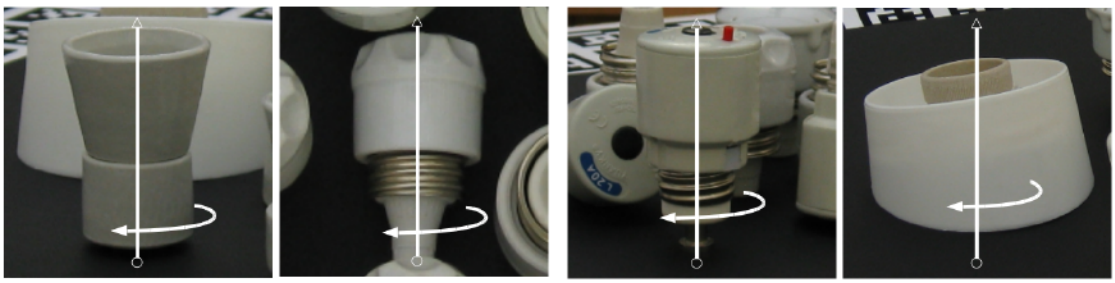


Figure 3.11: Contour generation to predict the poses of seven objects from the T-LESS dataset. Top row: *RoI* after object detection for objects 9, 3, 5, 21, 18, 7, and 6 in different orientations. The middle row shows the contour (green line) of the object and distance (yellow lines) of each pixel relative to the centroid pixel of the bounding box. The bottom row shows the distance signal after normalization that will be compared in our database to find the optimal match.



(a) (b)

Figure 3.12: Example of symmetries. (a) shows Obj_{15} (left) and Obj_1 (right), having symmetries around the y axes. Rotation from 0 to 180 degrees will generate the same images, which will cause ambiguities. (b) shows asymmetrical objects Obj_4 (left) and Obj_{18} (right), which generate different views while rotating horizontally.

Addressing object symmetry issues

Most T-LESS dataset objects have multiple symmetries that make pose estimation difficult. Likewise, other objects present certain details in their physical structures, such as small holes or markers, that make an object not completely symmetrical. However, these details can only be observed through a camera from a limited range of perspectives, which can easily be hidden by external occlusions or self-occlusions. These object symmetries or details are critical for industrial applications because

they are often used for certain specific actions, e.g., fitting a hole into its correct final position, either manually by an operator or automatically by a robotic arm. Therefore, it is necessary to identify the poses of these symmetries. To handle this problem during neural net training, we first define an initial orientation $[\phi_x, \phi_y, \phi_z]$ for each object in order to control the ranges of its 3D orientation from a reference during the rendering process. Then, we perform a visual perception analysis to find the symmetries along the axes of each object. An important step is to avoid symmetry on an axis for each object. For example, an object may have a reflection symmetry on the x-axis in an interval $\tau = [0, \pi]$. This can be constrained entirely by assigning zero to this axis $[0, \phi_y, \phi_z]$. As shown in Fig. 3.12a, both objects have symmetries around the y-axis, and therefore we stop rotation on that axis by assigning $[\phi_x, 0, \phi_z]$, where ϕ_x and ϕ_z can denote a rotation in a range $[0, \pi]$. In this case, n ranges $\tau_1, \tau_2, ..., \tau_n$ can be assigned to each object. Another example is rotational symmetry, where the object has isometries that preserve orientation in the Euclidean space. Avoiding symmetrical images will reduce the number of elements in our database, as well as the contour matching time. The rotational angles of the object are defined from 0 to 2π on all three axes. Therefore, when there is symmetry in a respective axis, we assign this angle a zero value, which means that no rotation is performed on this axis.

3D Location estimation

We measure the 3D translation of the object by using the center pixel of the object's bounding box. This box is represented by the smallest rectangle containing the RoI after the segmentation stage, denoted by the vector $[x, y, w, h]$, where x and y are the coordinates of the center of mass in the labeled region, and w and h are width and height of the bounding box, respectively. Every pixel of an object defined by its bounding box has a depth value corresponding to its 2D location. The distance of the object from the camera is estimated using intrinsic and extrinsic camera parameters. The location of a projected 3D point is obtained by correlating the 2D and the depth data of a pixel, as described below.

Firstly, the points of interest associated with 2D and depth information are extracted from the bounding box. Pixels of no interest are assigned zero values, while each important pixel is kept and assigned an index value to iterate over the columns and rows, which are the outer loop in the vertical axis ℓy and the inner loop in the horizontal axis ℓx, respectively. Inside the inner-loop, we compute the iteration and linearize the address space for $\mathcal{D}$. An independent depth value is extracted by applying $\mathcal{D}[u, v]$ as a pointer to indicate the value.

We assume the 2D coordinate space of the image as a continuous 2D space, where

a real-value (float) representation of the projected $\mathcal{D}_{x,y}$ is constructed by considering both δx and δy. The result in meters is computed from the above-mentioned depth value and a previously extracted scale value $scale_{factor}$. We estimate the translation $\mathcal{T}r_{x,y,z}$ of the object in 3D with all the pixels that segment the object. The formulation is given below:

$$\mathcal{T}r_z = \frac{\mathcal{D}[u,v]_{centroid} \in RoI_{obj}}{scale_{factor}} \tag{3.11}$$

$$\mathcal{T}r_x = \frac{(\mathcal{D}[u] - \delta x)\mathcal{T}r_z}{f_{length}} \tag{3.12}$$

$$\mathcal{T}r_y = \frac{(\mathcal{D}[v] - \delta y)\mathcal{T}r_z}{f_{length}} \tag{3.13}$$

3.3 Validation database and metrics

In this section we describe the public T-LESS dataset and our 3D printed T-LESS dataset used in the experiments, as well as the evaluation metrics used to validate our results.

Public T-LESS dataset

We evaluate our method on the T-LESS dataset [125], which provides a challenging and very realistic industrial scenario. This dataset contains 20 scenes (video sequences having a total of 504 frames) with multiple configurations of 30 objects captured on an electric turntable from different camera perspectives.

The authors have made available training and testing sequences, and a non-textured 3D CAD model for each object. They captured the scenes with three different sensors: Kinect (RGB-D), PrimeSense (RGB-D), and Canon (RGB). The T-LESS objects have no texture or color, and they are very similar to each other. Furthermore, all these objects show symmetries in specific axes, which causes pose ambiguity and often fails even SOTA approaches.

We tested our method on the images captured by the Kinect sensor, but images captured with the PrimeSense or Canon sensor can also be used. The scenes contain four or more object in a simple configuration. However, they are challenging for vision-based methods due to occlusion, uniform background, lack of texture, and the symmetry problem. The complex scenes can contain more than 15 objects, including duplicate objects, heavy occlusion, symmetry, shape similarity, and objects not usually seen. We tested on all sequences to evaluate the effectiveness of our method.

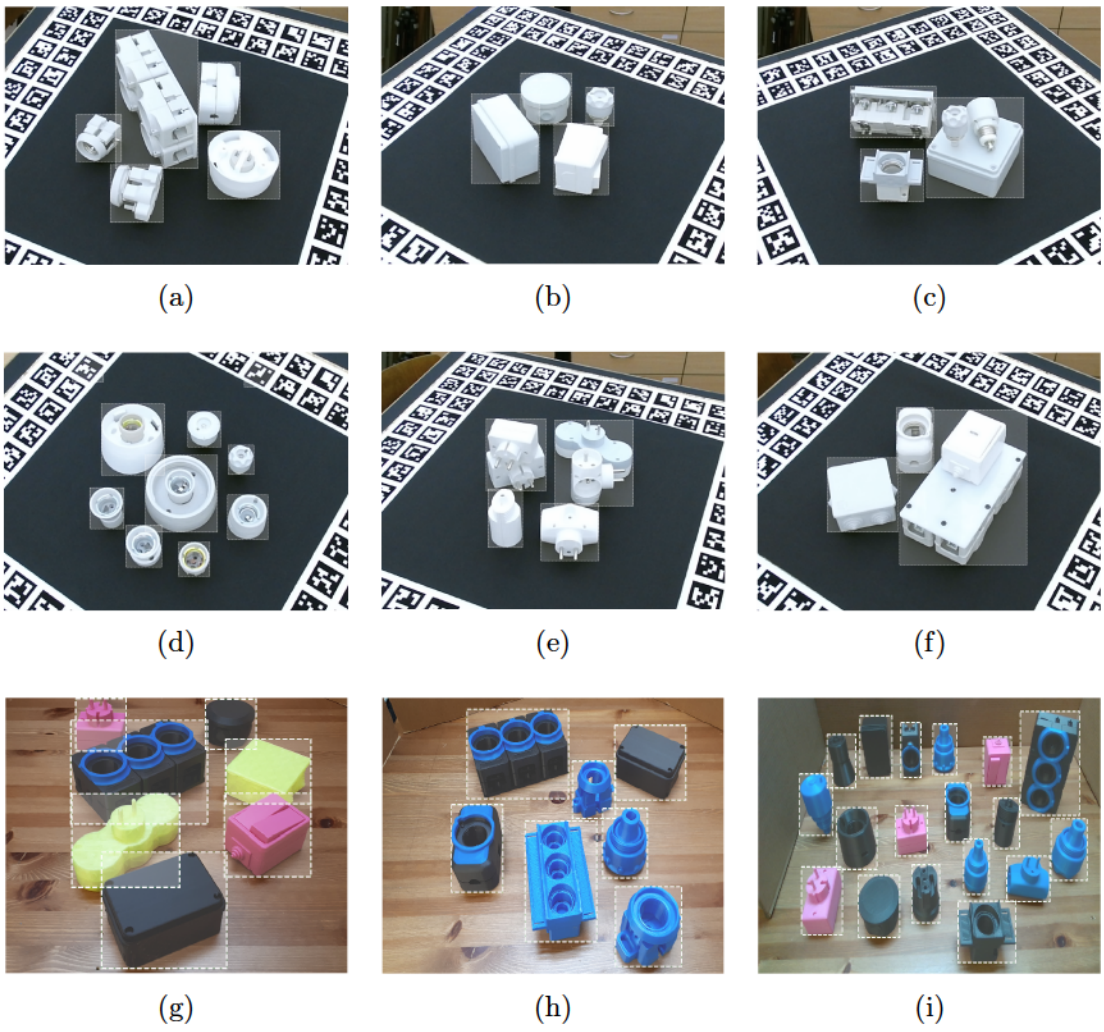


Figure 3.13: Estimated bounding box obtained from our object detection method. (a) and (b) show white bounding boxes precisely overlapping each object. (d) shows objects of different sizes with their respective *RoI*. (c), (e), and (f) demonstrate challenging configurations, where clustered objects are difficult to detect. (g), (h), and (i) show estimated bounding boxes in the test scenes using our own 3D-printed T-LESS objects.

Our 3D printed T-LESS objects

For the experiments, we acquired 30 scenes of our 3D-printed objects with different backgrounds and lighting conditions. Our objects have complex designs and are suitable for evaluating our pipeline's reliability. Some objects were partially occluded Fig. 3.13(g) as well as separated Fig. 3.13(h) and Fig. 3.13(i). In these cases, depth images were not available during testing. Therefore, we used a different segmentation stage, but later the same recognition process using HOG features and invariant moments was applied. A color deconvolution, called Haematoxylin-Eosin-DAB (HED) [128]

color space, is applied on the RGB image. The transformation is achieved by using a standard 3x3 matrix $\mathbf{M}_R$ that separates three-channel and two-channel images. The row vectors $\vec{H}_R$ (Haematoxylin), $\vec{D}_R$ (DAB), and $\vec{E}_R$ (Eosin) are the vectors for three color channels. Color deconvolution is able to segment shiny material objects in a specific HED component by using a cut-off value. Furthermore, we convert the input image to CIE-LAB color space to detect translucent parts. L describes lightness, and A and B are the chromaticity coordinates. The histograms in both HED and CIE-LAB indicate a cut-off point to segment target objects from the background. We proceed to normalize each RoI to 60x60 and recognize the objects using HOG features (cell size 6×6 and histograms with 4 bins) and invariant moments.

3.3.1 Validation metrics

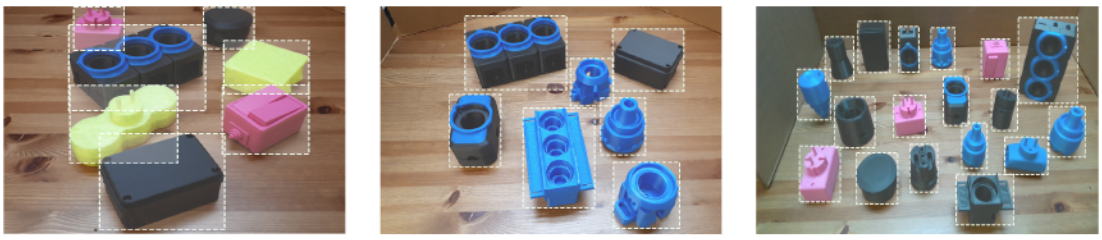
We follow the standard quantitative metrics to validate our algorithms on object detection and pose estimation. The system's performance for object detection is measured in terms of intersection over union (IoU). This metric describes the intersection between the predicted bounding box of the target object and the ground truth bounding box. Objects are considered properly detected in the image if IoU > 0.7. We also measure the Precision, Recall, and Specificity. For the 6D pose, we evaluate the visible surface discrepancy (VSD) score. This metric represents an ambiguity-invariant function to determine the error over the visible part of the model's surface. More specifically, the distance of the visible depth surface between the estimated and ground truth is calculated. The following equation defines the visible surface discrepancy:

$$e_{\text{VSD}}(\hat{\mathbf{P}}, \bar{\mathbf{P}}; \mathcal{M}, I, \delta, \tau) = \underset{p \in \hat{V} \cup \bar{V}}{\text{avg}} \; c(p, \hat{D}, \bar{D}, \tau), \tag{3.14}$$

where $\hat{V}$ and $\bar{V}$ are 2D masks of the visible surfaces of $\hat{\mathcal{M}} = \hat{\mathbf{P}}\mathcal{M}$ and $\bar{\mathcal{M}} = \bar{\mathbf{P}}\mathcal{M}$, respectively. $\hat{D}$ and $\bar{D}$ are distance images that we get by rendering $\hat{\mathcal{M}}$ and $\bar{\mathcal{M}}$. A distance image stores at each pixel p the distance from the camera center to the closest 3D point $\mathbf{x}_p$ on the model's surface that projects to p. δ is defined as a tolerance score used for estimating the visibility masks. $c(p, \hat{D}, \bar{D}, \tau) \in [0, 1]$ is the matching cost at pixel p.

3.4 Results

In this section, we report the validation results of our object detection, class recognition and 3D pose estimation methods.



(a) (b) (c)

Figure 3.14: Estimated bounding boxes in the test scenes using our own 3D-printed T-LESS objects

3.4.1 Object detection

We introduce a method for locating objects based on optical images and depth maps to predict the object location in an image. Our method is different from other approaches used, in particular CNN architecture such as SSD or Fast-RCNN, which needs to go through training before applying the trained model to locate the object. In contrast, our segmentation and localization technique is unsupervised, based on clustering and morphological operations. As will be discussed in the next section, supervised learning begins at the object recognition stage. We validate our object detection accuracy by computing the intersection over the union of the ground truth bounding box and our predicted bounding box of the target object. We can segment objects quickly without relying on learning to perform this task. However, in high occlusion situations, the accuracy can be affected. We performed various tests to determine the reliability of our object localization method for different scenes. We found that it is generally easier to localize objects in scenes with uniformly distributed objects, even when there are partial occlusions between the objects, e.g., Object A overlaps Object B by 10%. This is because our method performs clustering that allows us to distinguish two objects using their depth maps. The IoU metric describes the intersection of the predicted bounding box of the target object (RoI) and the ground truth bounding box. Objects are considered detected correctly in the image if IoU > 0.7. After that, object recognition is performed using histograms and invariant moments. We tested our object detection method in different scenarios, as shown in Table 3.1. Once the bounding box is detected, we determine the class of the object. The bounding area is passed to the recognition stage. We use two different parameters to extract the HOG characteristics: 10×10, and 15×15 pixels per cell, in addition to the invariant moments. In the next section, we analyze how different HOG parameters contribute to object recognition.

Table 3.1: Detection accuracy comparison using T-LESS test scenes: Sc_2, Sc_3, Sc_4, Sc_5, Sc_6, Sc_7, Sc_8, Sc_{11}, and Sc_{14}. Objects are considered properly detected in the image if IoU > 0.7. The last column reports the detection accuracy of each scene. The last row reports the average detection accuracy of all scenes.

Scene ID: Obj ID	HOG_{15x15}+HU	HOG_{10x10}+HU	Acc. (%)
02: 5	54.3	92.5	92.5
03: 5	17.8	29.0	
03: 9	11.2	17.5	
03: 12	18.9	27.7	
03: 18	15.1	26.5	25.2
04: 5	22.3	40.1	40.1
05: 9	55.6	77.8	77.8
06: 12	31.2	52.5	52.5
07: 18	47.8	67.5	67.5
08: 21	29.4	55.3	
08: 22	32.7	53.2	
08: 23	42.1	69.5	59.3
11: 5	41.8	66.0	
11: 9	43.6	67.5	66.7
14: 23	51.9	70.2	70.2
Avg.	34.38	54.16	61.3

3.4.2 Class recognition

Our object recognition method is not CNN based. Nevertheless, we can demonstrate that our results for object recognition are comparable with other methods. We generated training images of each object through a random rendering process, which does not require tedious manual labelling. Our approach provides a more convenient process for industrial applications. For example, if a new industrial part is added into the pipeline, the system only needs to render the CAD model of the new part from different perspectives to revise the trained model and there is no need to manually capture and collect an exhaustive training set of the new part.

Our method is not computationally expensive. However, calculating HOG features of an image can be an expensive operation, especially for high resolution images. To address this issue, we first extract the *RoI* and then normalize the region to an acceptable scale suitable for real-time tasks. We conducted a study on the effect of different parameters, such as image resolution and the initial HOG parameters (like the number of orientation bins and the number of pixels per cells), on the execution time and accuracy (Fig. 3.8). We observed that the final concatenated feature vector depends on the initial parameter to obtain relevant information. Also, the number of orientation bins and number of pixels per cell define the size of the feature vector, which may contribute to the computational time. We carried out several experiments with the 30 objects to determine the ideal values that allow us to maintain an accuracy above 90% and the shortest possible processing time, as shown in Fig. 3.8. We trained four different classifiers (SGD, PERCEPTRON, PASSIVE-I and PASSIVE-II) to compare their performances with different HOG parameters.

3.4.3 Object pose estimation

Once the class of the object is obtained in the recognition stage, it is sent to the pose estimation stage to perform the matching process. To validate the object pose estimation, a contour matching strategy is incorporated into the pipeline. All the contours from different views are extracted individually to create our database. We 3D printed 30 T-LESS objects and created 30 datasets, one for each object. Once the contours are extracted, the distance from the origin of the *RoI* centroid to each pixel on the edge is calculated, starting the measurement from a reference pixel on the edge. The same process is applied to the input images. Similarities are measured in the matching process to detect the best pose candidates. The matching process is through DTW, which is a scale invariant method. This algorithm allows for the finding of similarities between multivariate temporal sequences. During object detection, it

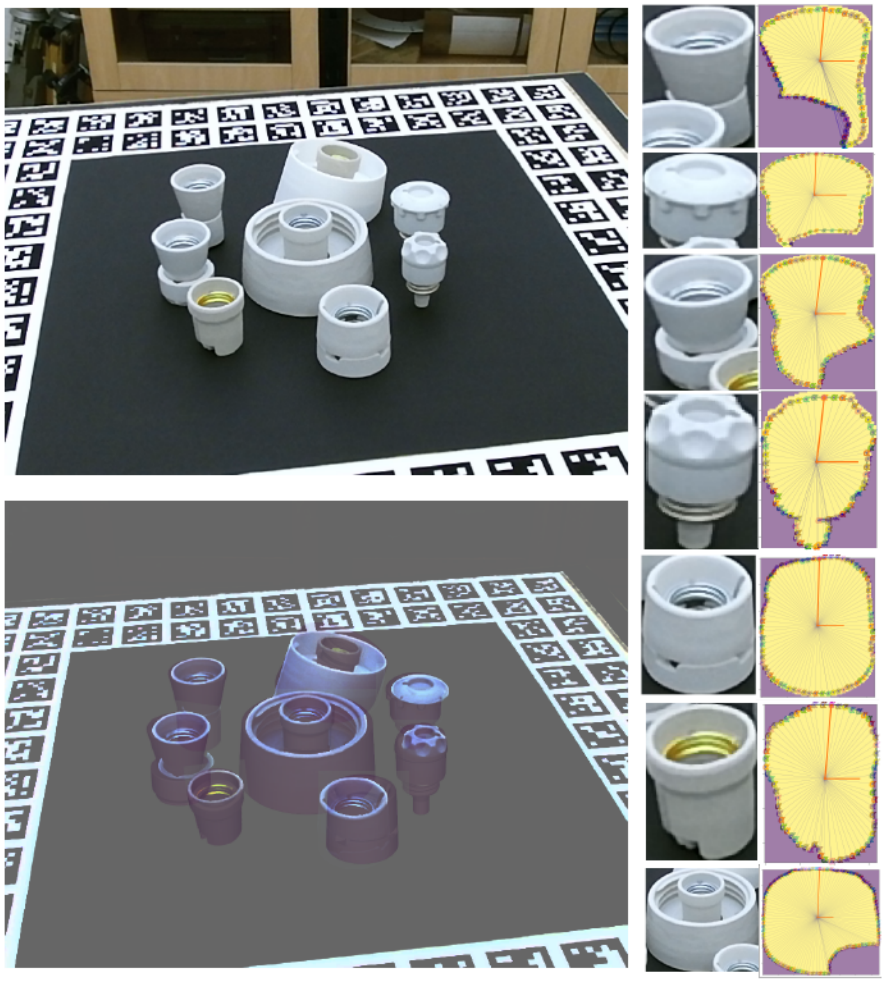


Figure 3.15: Qualitative results of our pipeline on T-LESS scene. Left top: Input image with multiple objects, including partially occluded cases. Right: segmented mask and contour generation for each mask. Left bottom: Qualitative pose estimation results using our pipeline. Candidate poses (blue color) for each detected object overlap the input image.

may happen that the size of the *RoI* is different for the same object with the same orientation but in different positions, which would cause scaling variations, e.g., the object is nearer or farther from the camera. In such cases, the contour measurement will be made longer or shorter depending on the position of the object. At the same time, the signal will be of different lengths in our database. However, DTW guarantees that our strategy is invariant to object scaling, because this method synchronizes the durations of both signals by matching features over time. Our results are shown in Table 3.2, which shows that our method outperforms other approaches.

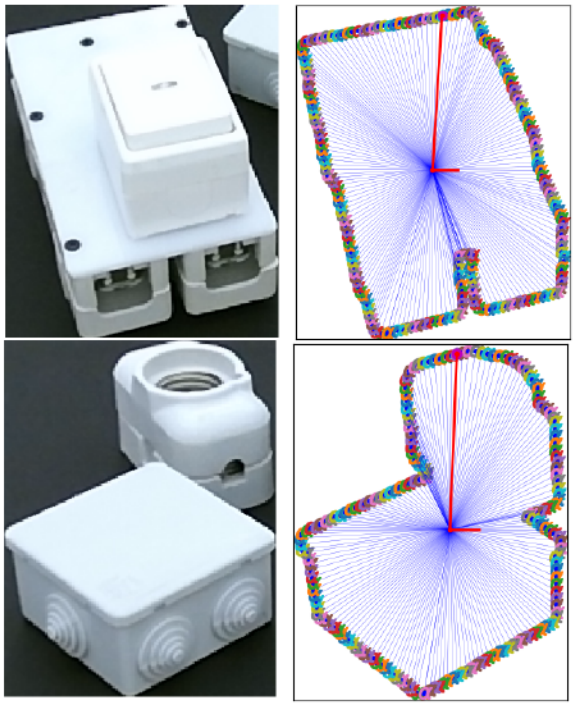


Figure 3.16: Failure cases: Our object segmentation pipeline fails for both cases. Top: it cannot separate Obj ID 26 and Obj ID 8; the pose can be reasonable for Obj 8 but not for Obj 26 since its contour area is limited. Bottom: despite Obj 5 and Obj 28 sharing the same mask, DTW can still provide an acceptable result for Obj 5 but not for Obj 28.

3.5 Result analysis and comparison with related work

In this section, we analyze how recognition is influenced by different initial parameters of the HOG algorithm, and the effectiveness of including invariant moments as features. We then discuss the influence of the rotation step criteria during training to obtain the final optimal pose.

3.5.1 Recognition results on T-LESS dataset

Our method demonstrates successful object recognition on scenes of the public T-LESS dataset. Although our model is trained with synthetic images, it can also predict the classes of real images, as shown in Fig. 3.13. In most scenes, our method worked reasonably well in detecting objects in uniform and irregular backgrounds. Moreover, specific camera angles yielded better results. This is because the improved lighting in these angles affects the depth maps, sharpening the features that the

image contains. Regarding object recognition, some objects were poorly recognized due to their similarity to others, e.g., Objects 1 and 2. The trained model could not distinguish these classes correctly because the HOG features and invariant moments for both objects are very similar. Object 5 was recognized in most of the different scenes, despite its similarity to Object 6. It was poorly detected happened when there was a significant occlusion, which sharpened its shape and features. These occlusions hide relevant information, and the only visible part of the object does not provide discriminative information. On the other hand, different objects that show the same symmetry from different perspectives can be misclassified.

3.5.2 Recognition results on our 3D printed T-LESS dataset

When using our 3D printed T-LESS dataset, the recognition results are shown in Table 3.5. We evaluated the performance of our algorithm in 30 scenes with different background and lighting settings compared to the public T-less settings. However, it should be mentioned that while our scenes contained fewer frames than the public T-LESS dataset, the objects have different colors and perspectives. This diversity of configurations in both datasets allows us to better validate the performance of our algorithm in different configurations for the same 30 different objects. Table 3.5 reports an average accuracy of 70.10% for our own printed T-LESS scenes. For these scenes, our algorithm demonstrates variable behavior in the detection of each object. Some objects are detected with high reliability, such as objects 4, 24, and 25 (accuracy above 80%), while others are detected with low accuracy, such as objects 3, 16, 6, 29, and 30 (accuracy under 70%). This can be due to various factors such as the color of objects, changes in lighting, or occlusion of important features. For example, Fig.3.13(g) Object 29 and Object 30 are black objects lacking visual attributes such as sharp edges and corners and are less noticeable if the lighting projected onto them is weak. In this case, the HOG features do not work effectively due to poor edge quality in the RoI. Poor detection performance can also occur on shiny material objects (Fig.3.13(g), Object 23 and Object 28) objects due to the incorporation of visual noise during image acquisition. In some cases, the segmentation stage can be affected by the poor quality of the depth image, consequently affecting the contour estimation and pose, as shown in Fig 3.16. The results also show that our model can handle partial occlusion in many cases. However, excessive occlusion continues to pose challenges and remains an open research problem.

3.5.3 Comparison with other approaches

We compare our method with other SOTA methods, specifically Sundermeyer [121] and Hajari [1]. [121] introduced a self-supervised training strategy based on autoencoders to estimate a 3D pose on RGB images, using synthetic views from a 3D CAD model. [121] first used a CNN architecture + SSD for object detection, obtaining 14.67% accuracy. Then, their results were improved to an average accuracy of 18.35% using RetinaNet architecture. [1] proposed a model adapting a global approach that recognizes an initial region of interest from an RGB image. The pose is obtained from the corresponding depth information by using a template-matching strategy on the point cloud based on surface normal and Fast Point Feature Histograms, which achieves an accuracy of 12.16%. Our method is comparable with approaches that perform pose estimation from optical images (Table 3.2). In our pipeline, the depth maps are only used for object segmentation at the initial stage. Our method outperforms most SOTA approaches, obtaining 24.23% on object recall with $err_{vsd} < 0.3$ on the T-LESS dataset. A comparison of results for pose estimation (Object 5 and Object 11) is shown in Table 3.3. There are some differences between authors with respect to the experimental settings used to obtain the final results. For example, in [123] and [1] the authors showed their results after using the ground truth bounding box (GT BBOX), meaning that the detection and recognition stages were not performed. In [121] the authors showed a comparison using Retina and SSD algorithms on RGB images with a basic ICP algorithm (optional) for refining depth data, which increases the computational time. In contrast, our reported statistics include all the stages of our pipeline from detecting to obtaining a final pose.

In Table 3, our method has a better average score than the others except Kehl *et al.*, and ours has comparable scores of 26.7 and 24.23 respectively. However, we observed that Kehl *et al.*'s method has better scores mainly from Obj ID 20 to 30, but its performance is reduced for the remaining objects. Our method performs better than Kehl et al. from Obj ID 1 to 20, scoring 27% and 16%, respectively. This means that our method performs better for a specific group of T-less objects. It should be noted that Obj ID 1 to 20 cover a large number of industrial objects with small shapes (ID 1-4, 13-15), medium with similar shapes (ID 5-6), and very similar shapes but different sizes (Obj 11 and Obj 12). On the contrary, objects from ID 20-30 are usually bigger than the previous ones, and have flat and square features. When we compare the computational time in Table 5, the enhanced version of Kehl *et al.*, and our method have a comparative fps rate of 12 and 8, respectively. Some approaches are very fast, e.g., Sundermeyer *et al.*, shows 42 fps, but their accuracy is very low

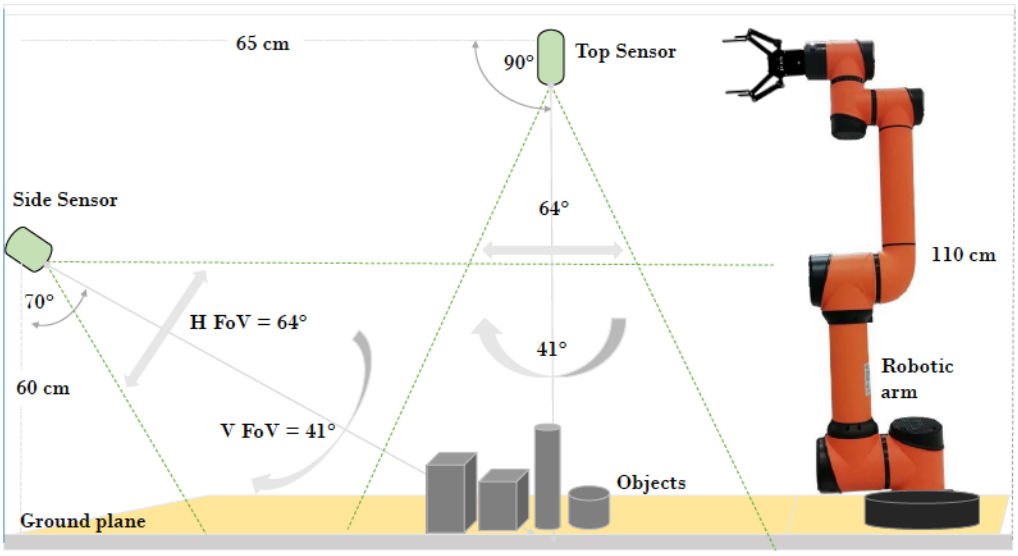


Figure 3.17: Manufacturing applicacion setup.

(14.67 and 18.35 on 6D SSD and 6D Retina algorithms respectively). As explained in Section 6.3, the processing pipeline is composed of many stages. Certain reported results do not include all the stages, e.g., Crivellaro *et al.*, Hajari *et al.*, and Pix2Pose (26.79%) reported their results after using the ground truth bounding box, which means that the detection and recognition stages were not included.

3.6 Parts manufacturing application

Our pipeline for object recognition and pose estimation was designed with a particular focus on an industrial robotic-arm pick-and-place application. The application aims to automate manual tasks in an automotive metal parts company. Typically, in these industrial environments, computer numerical control (CNC) often requires that drilling machines be loaded with heavy materials to produce a manufactured part. In our prototype, we used an AUBO robot manipulator and two RealSense sensors to capture images, as shown in Fig.3.17. All objects captured in our RGB-D scene were detected and recognized by our pipeline. The final pose of each object was recognized. The object was grabbed by the robotic arm and positioned in the final position. The current implementation runs on a desktop with an Intel Core i7-7000 CPU 2.80 GHz 4 Core. The execution time for a single object is 8 fps, surpassing our previous approach [1] and others, as shown in Table 3.4. The frame rate might vary when more instances are present in the scene.

3.7 Conclusion

We propose a semi-supervised strategy to obtain the pose of a texture-less object placed in a complex scene. Our system is partially trained on synthetic images using the 3D CAD model of the target object. We locate the *RoI* that delimits the object's structure through an unsupervised segmentation technique. An intelligent neural net is trained to recognize objects in different perspectives using features of Histogram Oriented Gradient (HOG) and invariant moments. The estimation of the object's pose is based on a matching technique using the Dynamic Time Warping method to compare vectors of different sizes without losing representative characteristics of the features. Our pipeline, integrating vision-based and learning-based approaches, demonstrates competitive performance when compared to existing methods on texture-less objects, and is suitable for real-time applications. In future work, we will explore optimization to further improve the method's time and accuracy performance.

Table 3.2: Comparing the accuracy of our method with the baseline methods on T-LESS dataset. We use the object recall for $err_{vsd} < 0.3$ metric on all test scenes.

	[121]		[1]	[5]	[129]	Pix2Pose	Ours
Id	6D SSD	6D Retina	FPFH			+GT 2D BBs	DTW
1	5.65	8.87	4.16	8.00	7.00	12.67	16.33
2	5.46	13.22	5.55	10.00	10.00	16.01	14.41
3	7.05	12.47	4.86	21.00	18.00	22.84	17.64
4	4.61	6.56	3.47	4.00	24.00	6.70	22.24
5	36.45	34.8	16.20	46.00	23.00	38.93	51.60
6	23.15	20.24	13.88	19.00	10.00	28.26	40.96
7	15.97	16.21	19.44	52.00	0.00	26.56	18.93
8	10.86	19.74	15.27	22.00	2.00	18.01	25.53
9	19.59	36.21	12.50	12.00	11.00	33.36	54.32
10	10.47	11.55	72.22	7.00	17.00	33.15	27.73
11	4.35	6.31	11.12	3.00	5.00	17.64	21.84
12	7.8	8.15	9.72	3.00	1.00	18.38	40.86
13	3.3	4.91	6.94	0.00	0.00	16.20	15.46
14	2.85	4.61	5.55	0.00	9.00	10.58	27.47
15	7.9	26.71	9.72	0.00	12.00	40.50	17.80
16	13.06	21.73	8.33	5.00	56.00	35.67	19.63
17	41.7	64.84	5.55	3.00	52.00	50.47	38.07
18	47.17	14.3	3.70	54.00	22.00	33.63	42.10
19	15.95	22.46	2.77	38.00	35.00	23.03	19.35
20	2.17	5.27	4.16	1.00	5.00	5.35	8.15
21	19.77	17.93	5.55	39.00	26.00	19.82	24.87
22	11.01	18.63	9.72	19.00	27.00	20.25	17.40
23	7.98	18.63	48.61	61.00	71.00	19.15	20.25
24	4.74	4.23	15.97	1.00	36.00	27.94	10.38
25	21.91	18.76	2.77	16.00	28.00	51.01	18.62
26	10.04	12.62	4.16	27.00	51.00	33.00	12.44
27	7.42	21.13	8.33	17.00	34.00	33.61	12.41
28	21.78	23.07	5.55	13.00	54.00	30.88	14.87
29	15.33	26.65	11.11	6.00	86.00	35.57	30.68
30	34.63	29.58	18.05	5.00	69.00	44.33	24.56
Avg.	14.67	18.35	12.16	17.07	26.70	26.79	24.23

Table 3.3: Pose estimation comparison results on T-less dataset, using Object 5 and Object 11 as examples.

Approach	T-Less	
	Obj. 5	Obj. 11
[123] + GT BBOX	0.19	0.21
[130]	0.69	0.69
[121] SSD - RGB	0.36	0.04
[121] SSD - RGB + Depth(ICP)	0.69	0.32
[121] Retina - RGB	0.34	0.06
[121] Retina - RGB + Depth(ICP)	0.76	0.35
[1] GT BBOX + FPFH (12 templates)	0.68	0.58
[1] FPFH (12 templates)	0.53	0.45
Ours	0.51	0.22

Table 3.4: Computational time comparison for single object pose estimation approaches

Approach	fps
[130]	0.2
[5]	2
[129]	2
[23]	4
[1]	5
Ours	8
[123]	10
[124]	12
[121]	42
[131]	50

Table 3.5: Object recognition results from our own 3D printed T-LESS dataset test scenes. Objects are considered correctly detected in the image if IoU > 0.7. Last row reports the average recognition accuracy of all objects.

1-10	1	2	3	4	5	6	7	8	9	10	
% Acc.	62.50	66.67	42.86	80.10	58.33	55.56	71.43	75.00	62.50	66.67	
11-20	11	12	13	14	15	16	17	18	19	20	
% Acc.	77.78	87.50	76.92	75.00	71.43	46.15		83.33	75.00	62.50	
21-30	21	22	23	24	25	26	27	28	29	30	
%Acc.	71.43	87.50	71.43	83.33	85.71	70.05	75.00	71.43	66.67	55.56	
Avg. 1-30											70.10

Chapter 4

Video analysis for local-area in-situ contexts

4.1 Introduction

The efficient classification of vehicles and the comprehensive analysis of their cargo have a significant role in modern transportation systems. Accurate insights into vehicular traffic, cargo distribution, and value assessment are crucial for effective highway management, route planning, and safety inspections. Through the examination of freight data, organizations can track the ongoing movement of goods, allocate these flows within the transportation infrastructure, and predict future freight patterns. These invaluable insights empower federal, provincial, and local authorities with the information necessary for informed planning and policy making.

According to Canada's transportation statistics [132], the trucking sector is a cornerstone of transportation within the supply chain. Due to the extensive network of roadways in Canada, trucks assume the foremost role in transporting goods across the nation. Additionally, the trucking sector serves as a pivotal mechanism for trade with the United States – Canada's primary trade partner. Road transportation overwhelmingly dominates the movement of freight and passengers in Canada [133]. The country boasts registration of over 25 million road vehicles and a comprehensive network spanning more than 1.1 million two-lane equivalent kilometers of public roads. In 2020, authorities revealed that within Canada's GDP sector related to transportation and warehousing, truck transportation emerged as the most commonly utilized mode for goods transport, constituting over 28.0% of the sector's activities [134]. Rail transportation followed at 11.4%, succeeded by air transportation and water transportation, both accounting for a combined 2.6%. Five leading commodities transported via trucks include machinery, manufactured goods, automotive products,

agri-food products, and chemical products. Concerning exports, a remarkable 97.3% of goods transported by trucks found their way to the United States, with 1.4% sent to Mexico and an equivalent percentage to other countries. Consequently, comprehending and vigilantly monitoring truck and vehicle activities is pivotal for gaining crucial insights into traffic dynamics. To address these imperatives, we introduce a novel analytics framework tailored to support vehicle classification and traffic analysis. Our framework combines advanced detection techniques, metadata exploitation, and validation strategies to offer precision and reliability in the classification of trucks and vehicles.

The fundamental premise of our framework centers on the need for real-time and historical understanding of traffic dynamics. By harnessing the power of machine learning and data analytics, we endeavor to decode the complexities of vehicular movements on highways, thereby enabling informed decision-making and streamlined operations. The ability to promptly access detailed information about detected vehicles empowers authorities to swiftly respond to potential incidents, ensuring heightened safety and security on the roads. Our framework goes beyond mere vehicle identification and enables us to ascertain the direction in which each truck or vehicle is traveling. This granular insight proves invaluable in optimizing traffic management strategies and resource allocation. We adopt a multi-tiered approach, encompassing automated validation during model training and on-site manual validation performed by certified agents. This validation strategy ensures that the framework's classification results are consistently accurate and dependable, even in dynamic and challenging scenarios. To ensure the sustained relevance and efficacy of our framework, we have engineered it for adaptability. Regular updates and refinements to the underlying models are integral to the framework's design philosophy. These iterative improvements are driven by a commitment to aligning with emerging data patterns and maintaining peak performance levels using an active learning stage.

In the subsequent sections, we provide details of our proposed framework, dataset, methodology, and validation processes. Through rigorous analysis and empirical evidence, we demonstrate the framework's effectiveness in enhancing vehicle classification accuracy and advancing the field of traffic analysis for improved highway management.

4.2 Literature Review

The effective management of highway systems and the accurate classification of vehicles are integral components of modern transportation infrastructure. Over the years,

researchers and practitioners have recognized the importance of harnessing advanced technologies to enhance the understanding of vehicular traffic dynamics and optimize resource allocation. In this section, we review relevant literature that underscores the significance of vehicle classification, traffic analysis, and the utilization of metadata in highway management.

4.2.1 Vehicle classification and traffic analysis

Vehicle classification is a foundational element in traffic engineering and management. The ability to categorize vehicles based on various attributes such as size, weight, and purpose is essential for optimizing road infrastructure, route planning, and safety inspections. Traditional approaches to vehicle classification often relied on manual observation and limited sensor data, which posed challenges in accuracy and scalability. Sensor networks have been used to achieve traffic intelligence and optimize road infrastructure [135]. Technologies are typically categorized into intrusive sensors, non-intrusive sensors, and off-roadway sensors. Intrusive sensors encompass various technologies like magnetic detectors, pneumatic road tubes, piezoelectric devices, and inductive loop systems [136–138]. These are embedded into the road surface using techniques like saw-cutting or hole installation. However, intrusive sensors might be susceptible to external magnetic interference, while also being less suitable for capturing information over larger areas. Non-intrusive sensors, on the other hand, offer alternatives like vision systems, microwave radar systems, and detectors based on infrared and ultrasonic technologies [139]. These sensors are installed overhead on roadways or roadsides, proving advantageous for covering significant traffic areas due to their non-invasive nature. This characteristic makes non-intrusive sensors a preferred choice for monitoring expansive traffic regions [140–144]. Recent advancements in machine learning and computer vision have spurred a paradigm shift in vehicle classification. Automated methods, such as deep learning techniques, have demonstrated remarkable proficiency in accurately identifying and categorizing vehicles in real-time [145–149]. These methods not only alleviate the limitations of manual classification but also enable a more comprehensive analysis of traffic patterns, leading to informed decision-making and improved highway management.

4.2.2 Active learning strategies

Active learning can be a crucial strategy for machine learning, particularly in the domain of object detection, due to its capacity to optimize model performance with only a small amount of ground truth data. Annotated object detection datasets are labor-

intensive to curate, and active learning is a powerful technique used to intelligently select the most informative data samples for annotation. By selecting examples that the model finds challenging, active learning ensures that the training process focuses on refining areas of uncertainty, thereby enhancing model generalization.

The authors of [150] define active learning as a core-set selection problem in which a subset of examples from a dataset are chosen for training a model such that the model will give competitive results on the remaining examples. They utilize the geometry of the dataset to provide a bound on the average core-set loss over any subset of the dataset and select the subset that minimizes the bound for image acquisition. A hybrid entropy-based sampling approach based on class imbalance metrics was proposed in [151], where oversampling of minority classes and undersampling of majority classes was dynamically accomplished via the class imbalance metric. A deep-learning method considering the training set loss was implemented in [152], where a training set loss-based weighting algorithm was used to balance Average Precision (AP) between object categories to improve the mean Average Precision (mAP). The authors of [153] addressed the issue of global instance-level redundancy by proposing a hybrid framework combining both instance-level and image-level redundancy removal, while simultaneously ensuring class balancing to increase diversity across images. They managed to achieve SOTA results on widely used research datasets.

4.3 Data

4.3.1 Data overview

The traffic videos were recorded at various locations across British Columbia, Canada. The videos encompass a diverse range of traffic scenarios and vehicle types, contributing to the comprehensiveness and reliability of our analysis framework. Camera view configuration was carried out strategically in such a way that vehicles can be viewed towards one or more directions of interest for each location. Note that the camera position and camera view are stationary for each study. The recordings were produced during the day and night, capturing multiple weather conditions. The input data typically includes 24-hour videos recorded for 1-3 days, and the frame rate varies from 25-30 fps with multiple camera resolution. An overview of the different camera locations and classes is shown in Table 4.1 and Table 4.2, respectively.

Figure 4.1: Images from our dataset with ground truth bounding boxes.

Table 4.1: Vehicle Classification Labels and Descriptions

ID	Label	Vehicle Type	Description
1	lt	Light Truck	Single rear-axle light trucks used for various commercial and utility applications.
2	tronly	Tractor Only	Tractor unit without an attached trailer or payload, often used for hauling larger trailers.
3	trchass	Tractor + Chassis	Tractor unit coupled with a chassis for versatile transport applications.
4	trflat	Tractor + Flatbed	Tractor unit paired with a flatbed trailer for transporting bulky or oversized cargo.
5	trtrail	Tractor + Trailer	Tractor unit hauling a standard trailer, common in long-haul transportation.
6	trcont	Tractor + Container	Trucks designed for transporting shipping containers in intermodal freight transport.
7	trtank	Tractor + Tanker	Trucks equipped for transporting liquids or gases in specialized tanker trailers.
8	const	Construction Trucks	Trucks for construction tasks, including dump trucks, cement mixers, and more.
9	waste	Waste	Vehicles used for waste management, garbage collection, and recycling operations.
10	o	Other/Single Unit 3 Axle	Miscellaneous single-unit vehicles with three axles.
11	vp	Van/Pickup + DG Placard	Vans and pickups carrying Dangerous Goods (DG) placards for hazardous materials.
12	trreefer	Truck with Refrigeration Unit	Trucks equipped with refrigeration units for transporting temperature-sensitive goods.
13	rvc	RVs and Campers	Recreational vehicles (RVs) and campers for leisure travel and camping.
14	bus	Buses	Various types of buses, including public transportation, school buses, and long-distance coaches.
15	sv	Small Vehicle	Cars, motorcycles, sedans, SUVs, and other personal vehicles.

4.4 Proposed framework

4.4.1 Data preprocessing

Manual sampling videos with significant frames

Initially, we tested using pre-existing models that can detect cars and trucks. These models are often pre-trained on public large datasets like COCO (Common Objects in Context) and show acceptable performance in detecting vehicles in many cases. However, our study noted that these existing models often misclassified trucks or did not detect them efficiently because of the diversity of the cameras' field of view and the scene complexity. Many scenes can present challenging cases such as low, moderate, or high occlusions, as well as crowds of vehicles grouped in close proximity. These difficulties necessitate building a more robust system to obtain much better

Table 4.2: Highway Camera Locations for Vehicle Analysis Videos

Site	Directions	Resolution	FPS
1900 Blk Capilano Rd	NE/SW	960×540	15
Hope Truck Stop Hwy 1 Ramp	E	720×480	30
Cole Rd Hwy 1 Ramp	NE/SW	720×480	30
Hwy 1 at Peterson Creek	E/W	720×480	30
Kamloops Break Check	E	720×480	30
Kamloops CVSE	W	720×480	30
Knutsford Rest Area	N/S	960×540	15
Main St and Cotton Rd	E/W	960×540	15
Marshall Rd and Peardonville Rd	E/W	960×540	15
Queen St and Wheel Ave	N/S	720×480	30

performance in the detection, classification, counting, and directional estimation of vehicles. Therefore, we consider selecting frames from multiple videos showing representative and diverse data for annotation purposes. We noticed a greater frequency of small vehicles (class 15) and Vans/Pickups (class 11) than the remaining classes. Consequently, we primarily focused on frames containing instances from classes 1-10 and 12, which correspond to truck vehicles. Within these frames, we proceeded to annotate all other vehicles from the remaining classes (classes 11 and 13-15) if they were present. The selected frames were separated into n batches per each video location for data labeling.

Data annotation

We utilize a crowdsourcing environment to distribute the annotation task among multiple labelers and reviewers. Proper quality control mechanisms and clear annotation guidelines are essential. To achieve this, n batches are divided among multiple annotators to work simultaneously. At this stage, labelers followed clear guidelines and consistent quality control to ensure uniform annotation. We started by annotating a smaller subset of the dataset and gradually expanded as needed. Our approach involved distributing annotation tasks among six annotators and leveraging four specialized reviewers with expertise in vehicle domains for approval for two weeks. Annotations rejected by reviewers prompted refining by the assigned reviewer. This process ensures data accuracy and domain-relevant insights. Following the annotation stage, we built our baseline dataset with a diverse collection of 2,867 images. Within this dataset, 93 images were categorized as null (devoid of any discernible objects of in-

terest), and 5,995 annotations were meticulously curated. With an average of 2.1 annotations per image, the dataset spanned 15 distinct classes. The images in the dataset exhibit an average size of 0.35 megapixels, with the median image resolution being 720x480 pixels. Notably, the most prevalent class is the sedan (sv) with 2,011 instances, followed closely by the van/pickup (vp) with 932 instances, then the trailer with a trail (trtrail) with 900 instances. The remaining classes contain the following annotations: trflat (489), const (351), trreefer (280), lt (251), trtank (235), rvc (128), trcont (117), bus (111), tronly (94), o (59), and waste (37). The dataset offers a rich variety of traffic scenarios, providing an ideal foundation for in-depth traffic analysis and research endeavors, as shown in Fig.4.1.

Our approach's active learning aspect enhances efficiency. The framework is particularly beneficial for processing extensive video datasets, as exemplified by our input data of up to 72-hours-long MP4 videos. Collaborative annotation efficiently manages large-scale data, ensuring accurate model training. Furthermore, our proposed pipeline is specifically designed to effectively handle continuous input data, making it well-suited for real-world scenarios.

4.4.2 Data augmentation

Data augmentation is a prevalent technique within Computer Vision that aims to expand both the size and variety of the training dataset. This practice becomes particularly pertinent in the context of object detection tasks, as encountered in our ongoing investigation, given the complexities introduced by issues like occlusion and shifts in viewpoint. To address the intricate matters of occlusion and viewpoint variability, this study has implemented the following data augmentation operations: hue -25° to +25°, brightness -30% to +30% , exposure -20% to +20%, noise up to 5%, random cutout, and horizontal flip.

4.4.3 Coarse level model selection

This scenario studies the assessment of four distinct object detection models: YOLOv5, YOLOv6, YOLOv8, and RT-DETR-L. All these models are tailored for the single-stage detection of objects. To comprehensively evaluate their performance, multiple variations of each model were employed. Subsequently, the study explored the efficacy of a two-stage approach using YOLOv8. In this approach, an initial YOLOv8 detector determined whether a given vehicle was categorized as a 'truck' or 'other,' where 'truck' maps the classes 1 to 10 and 12, and 'other' maps the classes 11,13,14, and 15. Then, we trained a subsequent classifier model to precisely decide a final cat-

egory. We describe below the different models that we compared during our selection process.

YOLOv5

Developed as an upgrade to YOLOv4, YOLOv5 has a couple of new features including the PyTorch framework and AutoAnchor algorithm. Its architecture makes use of technologies including the SPPF layer and several augmentations for increased stability, as well as a modified CSPDarknet53 backbone.

YOLOv6

The Meituan Vision AI Department unveiled YOLOv6, a cutting-edge object identification model. A PAN topology neck, a creative decoupled head using a hybrid-channel technique, an effective backbone made of RepVGG or CSPStackRep blocks, and a decoupled head are all included in YOLOv6. Improved quantization methods, such as post-training quantization and channel-wise distillation, outperform YOLOv5, YOLOX, and PP-YOLOE in terms of speed and accuracy. The EfficientRep backbone, Task alignment label assignment, creative losses, self-distillation, and quantization technique are notable components.

YOLOv8

YOLOv8 was introduced by Ultralytics in January 2023. It offers five scaled variants to support various eyesight activities. By combining features and context, the C2F module, which replaces the CSPLayer, increases accuracy. To increase accuracy, YOLOv8 uses an anchor-free model with a decoupled head for separate objectness, classification, and regression tasks. Bounding boxes are improved by CIoU and DFL losses, and small objects benefit from binary cross-entropy. In terms of semantic segmentation, YOLOv8-Seg excels. It is significantly more user-friendly because it supports CLI/PIP use.

RT-DETR

Impressive results have recently been obtained using DETRs, or end-to-end transformer-based object detectors. However, their high computational cost makes them impractical to employ and prohibits post-processing techniques like non-maximum suppression (NMS) from being fully utilized. Real-Time Detection Transformer (RT-DETR) was proposed as a remedy to this problem. In addition to improving query quality and enabling adjustable speed modifications without the need for retraining, RT-DETR

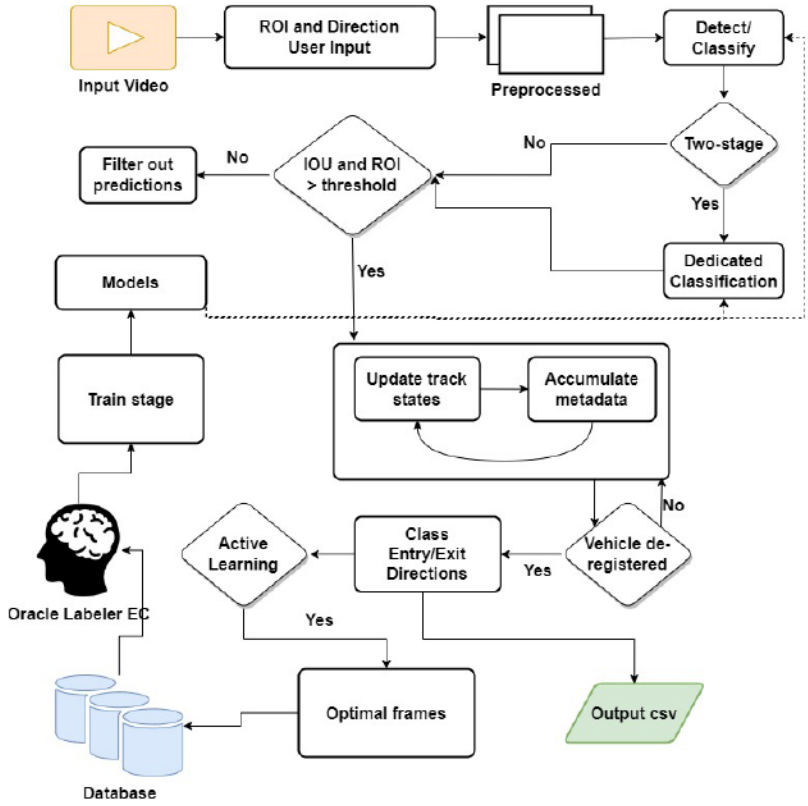


Figure 4.2: High-level overview of the closed-loop vehicle analysis pipeline

effectively manages multi-scale characteristics. It is also very user-friendly because it supports CLI/PIP use.

4.4.4 Metadata extraction from predictions

Metadata extraction provides the foundation for a robust traffic analysis system. We engineered a vehicle-tracking algorithm, illustrated in Fig. 4.2, capable of capturing per-vehicle class and class confidence estimations as well as entry and exit direction estimations. These are essential pieces of information that can provide pivotal insights into goods flow and route planning optimization. Our methods for obtaining these pieces of data are described below:

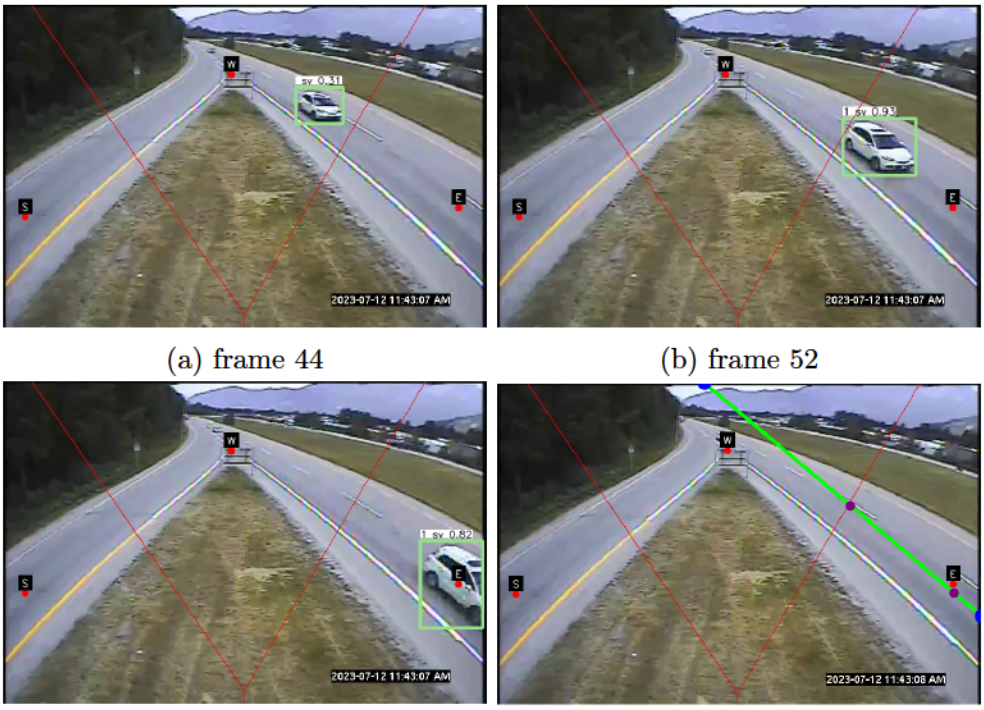
Vehicle tracking

We created both a single-stage and a two-stage pipeline for vehicle tracking. The single-stage pipeline utilizes a single YOLOv8 object detector trained on all vehicle classes. The two-stage pipeline utilizes a YOLOv8 detector trained with all vehicle classes merged into one vehicle class, and then a downstream YOLOv8-classifier model to get the fine-grained classification results. Both pipelines make use of the Norfair

tracking algorithm [154] to track detections across time. Each unique object track is associated with a vehicle instance, and once the object track is unregistered from the tracker it is counted as an instance of that vehicle's class. However, tracking failures showed in scenarios where multiple vehicles appear simultaneously. In such situations, the tracker's performance can be adversely affected, leading to difficulties in accurately identifying, distinguishing, and tracking individual vehicles, especially when occlusions, overlapping objects, or complex interactions occur among multiple vehicles. Because of the potential classification uncertainty in complex conditions, we adapted this tracker to enhance the accuracy of vehicle class prediction. Our approach involves calculating the cumulative average of confidence scores associated with each estimated class over a vehicle's lifespan. The final class is computed as $\arg\max_{c \in C} \frac{1}{n_c} \sum_{i=1}^{n_c} p_{i,c}$, where C is the set of vehicle classes being tracked, n_c is the number of occurrences of class $c \in C$, and $p_{i,c}$ is the confidence value of the i^{th} occurrence of class c. This enables us to make a more informed and robust final class prediction, especially in scenarios with multiple vehicles, by considering the historical confidence information. We recognize that different camera placements and streets may present unique challenges and variations in traffic patterns. In conjunction with the previously described algorithm, our system incorporates a user-defined region-of-interest polygon to filter out regions of the camera view that do not contain useful information for traffic flow analysis (such as faraway regions and side streets). We decided upon a polygon region of interest instead of user-defined lines across the road, such as those used in [155, 156], to specify where vehicle counting occurs as it can handle more complex scenes, such as intersections, to be analyzed.

Vehicle direction estimation

While vehicle counting is a common topic, vehicle flow direction analysis has not been sufficiently studied. We estimate vehicle entry and exit directions using the initial and final centroids ($\boldsymbol{c_i}$ and $\boldsymbol{c_f}$) of the vehicle, as well as user-defined direction coordinate points Q. We first bucket each image pixel $\boldsymbol{p}$ into direction zones via the following equation: $\text{zone}(\boldsymbol{p}) = \arg\min_{q \in Q} \|\boldsymbol{p} - \boldsymbol{q}\|_2$. We then build an image mask from these buckets, where its pixel values correspond to the index of the direction coordinate bucket they fell into. When a vehicle is unregistered from the tracker, we fit a straight line $y = m \cdot x + b$ through $\boldsymbol{c_i}$ and $\boldsymbol{c_f}$ and then compute the points where that line intersects the image boundaries. We then do a one-to-one association of the centroids to intersection points based on shortest relative distances. Finally, the pixel value of the image mask at each intersection point denotes the directions associated with the centroids, and thus the vehicle itself.



(a) frame 44 (b) frame 52

(c) frame 58 (d) direction

Figure 4.3: Example of direction estimation for a vehicle that was tracked. The purple points in (d) correspond to the initial and final centroid estimations, and the blue points correspond to the intersection of the fit line with the image boundaries. The red lines depict the direction zones. This vehicle would be classified as entering westbound and exiting eastbound

The mask is only computed once at the start, and then the intersection and direction association steps are computed each time a vehicle is unregistered. These calculations can be performed quickly, and we bypass the need to store all centroids across the lifespan of a vehicle, thus making it both computationally and memory efficient. Fig. 4.3 displays what this may look like during a processing run.

Algorithm 1 Active Learning Image Acquisition

Input: video frames $\{I_i\}_{i\in[n]}$
classes C
image budget b
instance similarity thresh T_{enms}
intra-class redundancy thresh T_{intra}
inter-class balancing thresh T_{inter}
video frame rate fps
seconds between samples s

Output: selected images ΔS for oracle label corrections

// simplifying $\{1,\ldots,n\}$ to $[n]$ for readability
initialize candidate images $S \leftarrow \emptyset$, class predictions $Q \leftarrow \emptyset$, confidence scores $P \leftarrow \emptyset$, image entropies $E \leftarrow \emptyset$, prototypes $T \leftarrow \emptyset$, and sampling rate $r \leftarrow 0$
for i in $[n]$ **do**
  $r \mathrel{+}= 1$
  Detect objects o in the image I_i, obtaining the bounding boxes $\{b_{i,k}\}_{k\in[o]}$, classes $\{q_{i,k}\}_{k\in[o]}$, and confidences $\{p_{i,k}\}_{k\in[o]}$
  if $o \neq \emptyset$ and $r \geq fps \cdot s$ **then**
    update $S \leftarrow S \cup \{I_i\}$, $Q \leftarrow Q \cup \{q_{i,k}\}_{k\in[o]}$, $P \leftarrow P \cup \{p_{i,k}\}_{k\in[o]}$, and $r \leftarrow 0$
    for k in $[o]$ **do**
      extract single-object image I_k from I_i using $b_{i,k}$
      compute feature embedding $\boldsymbol{f}_{i,k}$ from I_k using the CLIP zero-shot model
    end for
    update $E \leftarrow E \cup \{\text{ENMS}(Q_i, P_i, \{\boldsymbol{f}_{i,k}\}_{k\in[o]}, T_{\text{enms}})\}$
    compute prototypes T_i using equation 4.1 and update $T \leftarrow T \cup \{T_i\}$
  end if
end for
// function call holds lines 2-13 of Algorithm 2 in [153]
set $\Delta S \leftarrow \text{DiversePrototype}(S, T, b, T_{\text{intra}}, T_{\text{inter}})$
return the most informative images ΔS, where $|\Delta S| \leq b$

4.4.5 Active learning and label assist

We developed an active learning algorithm for object detection inspired by the diverse prototype algorithm described in [153], with pseudocode shown in Algorithm 1. We first sample frames *explicitly* containing detections every n seconds from the video.

This ensures all initial candidate images are detected, and the sampling rate can be changed depending on the expected speed of traffic flow. After candidate images have been sampled, we iterate over them, computing the feature vector of each detected object utilizing OpenAI's CLIP zero-shot image classification model [157]. CLIP takes a different approach to learning visual representations by utilizing natural language supervision and has shown respectable generalization capabilities. At this point, we have all we need to begin the diverse prototype algorithm.

The algorithm begins with Entropy-based Non-Maximum Suppression (ENMS), where the entropy of each image is computed using per-object-instance entropy calculations and cosine similarity metrics to reduce redundancy and to ultimately find a better representation of the information in the image. Then, prototypes are computed for each detected class in each image. The prototype for class c detected in image i is formulated as [153]:

$$\text{proto}_{i,c} = \frac{\sum_{k \in [t]} \mathbb{1}\left(c, c_{i,k}\right) \cdot \mathbb{H}\left(I_i, k\right) \cdot \boldsymbol{f}_{i,k}}{\sum_{k \in [t]} \mathbb{1}\left(c, c_{i,k}\right) \cdot \mathbb{H}\left(I_i, k\right)} \tag{4.1}$$

where $\mathbb{1}\left(c, c_{i,k}\right)$ equals 1 if $c = c_{i,k}$ and 0 otherwise, and $\mathbb{H}(I_i, k) = -p_{i,k}log(p_{i,k}) - (1 - p_{i,k})log(1 - p_{i,k})$ is the entropy of the k^{th} object instance, computed based on its confidence score. These prototypes significantly reduce the computational complexity when computing diversity metrics across multiple images.

Images are then sorted in decreasing order according to their entropy and are successively selected based on their intra-class and inter-class diversity across previously selected candidate images. The algorithm ensures that the most optimal, information-dense frames are extracted within the specified image budget, allowing for massive time savings in comparison to manual dataset annotation. Our application outputs the images and annotations in the YOLOv8.txt annotation format, allowing for quick drag-and-drop into most annotation software. This active learning framework automates almost all the manual labor usually required for annotating object detection datasets. Users no longer have to manually sift through hours of videos to find optimal frames and annotate thousands of bounding boxes afterwards. Only slight bounding box touch-ups and misclassification corrections are necessary, which require much less time to complete.

4.4.6 Model training

All models were trained using the hyperparameters in Table 4.3. Also, all the models were trained for up to 200 epochs with a batch size of 8 and an image size of 640x640. In order to compare the effectiveness of the models, we use precision, recall, and mean

Table 4.3: Training Hyperparameters

Hyperparameter	YOLOv5	YOLOv6	YOLOv8	RT-DETR-L
Initial Learning Rate	0.01	0.01	0.01	0.0001
Optimizer	Adam	Adam	Adam	Adam
Momentum	0.937	0.937	0.937	0.937
Weight Decay	0.0005	0.0005	0.0005	0.0005
Warmup Epochs	3.0	3.0	3.0	3.0

Table 4.4: Experimental results on validation baseline dataset

Model	Precision	Recall	mAP@0.5	mAP0.5-0.95	Inf. (ms)
YOLOv8n	0.839	0.796	0.882	0.788	3.8
YOLOv8m	0.875	0.825	0.886	0.798	6.3
YOLOv8l	0.887	0.808	0.881	0.795	16.8
YOLOv8s	**0.898**	0.805	0.876	0.793	25.9
YOLOv8x	0.874	0.793	0.875	0.791	46.6
YOLOv6n	0.656	0.587	0.633	0.549	2.0
YOLOv6m	0.654	0.672	0.710	0.614	14.3
YOLOv6l	0.603	0.591	0.595	0.513	27.7
YOLOv6s	0.757	0.637	0.726	0.627	4.6
YOLOv5n	0.876	0.787	0.880	0.786	1.9
YOLOv5m	0.878	0.807	0.881	0.793	5.2
YOLOv5l	0.846	0.802	0.884	**0.800**	7.7
YOLOv5s	0.844	0.818	0.885	0.793	2.6
YOLOv5x	0.876	0.828	**0.890**	**0.800**	15.3
RT-DETR-L	0.850	**0.829**	0.820	0.734	17.7

Table 4.5: Experimental results on validation baseline dataset after data augmentation

Model	Precision	Recall	mAP@0.5	mAP0.5-0.95	Inf. (ms)
YOLOv8n	0.830	0.828	0.881	0.788	3.2
YOLOv8m	0.865	0.816	0.875	0.796	17.6
YOLOv8l	0.858	0.837	0.885	0.807	27.4
YOLOv8s	0.881	0.786	0.877	0.789	6.3
YOLOv8x	0.882	0.811	**0.891**	**0.812**	41
YOLOv6n	0.792	0.756	0.820	0.727	1.1
YOLOv6m	0.843	0.810	0.865	0.774	7.7
YOLOv6s	0.850	0.808	0.877	0.786	2.4
YOLOv5n	0.86	0.787	0.872	0.774	1.6
YOLOv5m	0.862	**0.838**	0.880	0.798	3.2
YOLOv5l	0.863	0.822	0.890	0.805	4.7
YOLOv5s	0.881	0.816	0.875	0.787	1.7
YOLOv5x	**0.885**	0.831	**0.891**	0.805	15.2
RT-DETR-L	0.876	0.831	0.842	0.754	16.9

average precision as metrics. For training and evaluation, we divide the workload into two machines. Both workstations have NVIDIA GeForce RTX 2080 SUPER, processor Intel Core i7-10700KF CPU with 3.80GHz×16, and 64GB of RAM.

4.5 Experimental Results

4.5.1 Single-stage object detection results on baseline dataset

We conducted a comparative analysis of the models in Table 4.4. Among the evaluated object detection models, YOLOv8s demonstrates remarkable performance in terms of precision, achieving a value of 0.898. This indicates the model's proficiency in making accurate positive predictions. YOLOv8l follows closely as the second-highest performing model in terms of precision, with a precision value of 0.887. On the other hand, RT-DETR-L and YOLOv5x showcase a recall of 0.829 and 0.828. YOLOv8m also performs acceptably in terms of recall, with a recall value of 0.825. Notably, YOLOv5x also excels in mAP@0.5 and mAP@0.5-0.95, exhibiting values of 0.890 and 0.80, respectively. YOLOv8m emerges as the second-best model for mAP@0.5, showing a mAP@0.5 value of 0.886. Additionally, in terms of mAP0.5-0.95, both

YOLOv8l and YOLOv5l share the lead with a value of 0.800, further emphasizing their strong performance in detecting objects under varying overlap conditions.

4.5.2 Single-stage object detection results on augmented baseline dataset

In the realm of object detection, a rigorous examination of diverse models was undertaken encompassing an augmented version of our baseline dataset, shown in Table 4.5. YOLOv5x underscored its proficiency by attaining a precision score of 0.885. YOLOv5m achieved 0.838 in recall metric followed by YOLOv8l with a recall score of 0.837 (refer to Fig. 4.9). RT-DETR-L and YOLOv5x achieved both 0.381. The performance of the models was further accentuated by their mAP@0.5. Here, YOLOv8x and YOLOv5x shared a mAP@0.5 score of 0.891, showing their capability to harmoniously balance precision and recall at an IoU threshold of 0.5. Extended across the broader IoU of 0.5 to 0.95, YOLOv8x remained resolute with a mAP0.5-0.95 score of 0.812, a crucial factor with contextual significance depending on the application's exigencies. This factor underscores the trade-off between accuracy and efficiency. Notably, YOLOv6n provides faster predictions, showing an inference time of 1.1 ms, rendering it adept for applications requiring real-time object detection. YOLOv5n showcases swift object detection deployment with an inference time of 1.6 ms. Conversely, certain models reflect a relatively higher inference time, encapsulating the intricacies of their architectures. Noteworthy among these models is YOLOv8x, which, while attaining commendable recall and precision, is characterized by a comparatively longer inference time of 41 ms. Please refer to Figs. 4.4, 4.5, 4.6, 4.7, and 4.8 for quantitative and qualitative results of YOLOV8x.

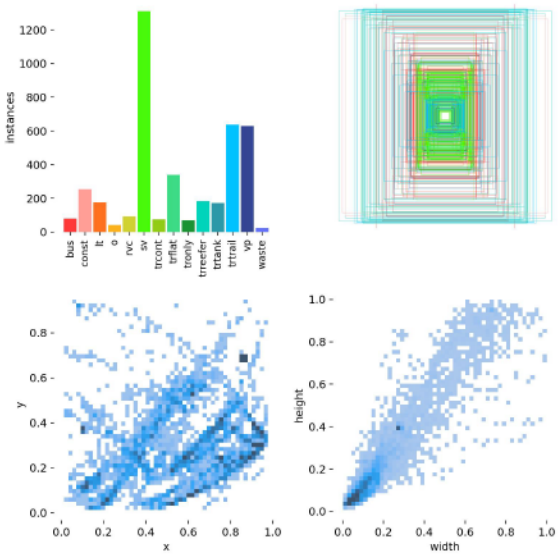


Figure 4.4: Top left subfigure shows the label distribution of the baseline train set, with each color representing a class. SV class contains most objects, followed by VP and TRTRAIL. The top right subfigure shows the size of the object bounding boxes in the dataset, and the coordinates of the centers of all object boxes are fixed at one point. The bottom left subfigure shows the distribution of the coordinates of the center points of the object's bounding boxes. The bottom right subfigure is a scatter plot of the corresponding width and height of the object's bounding box.

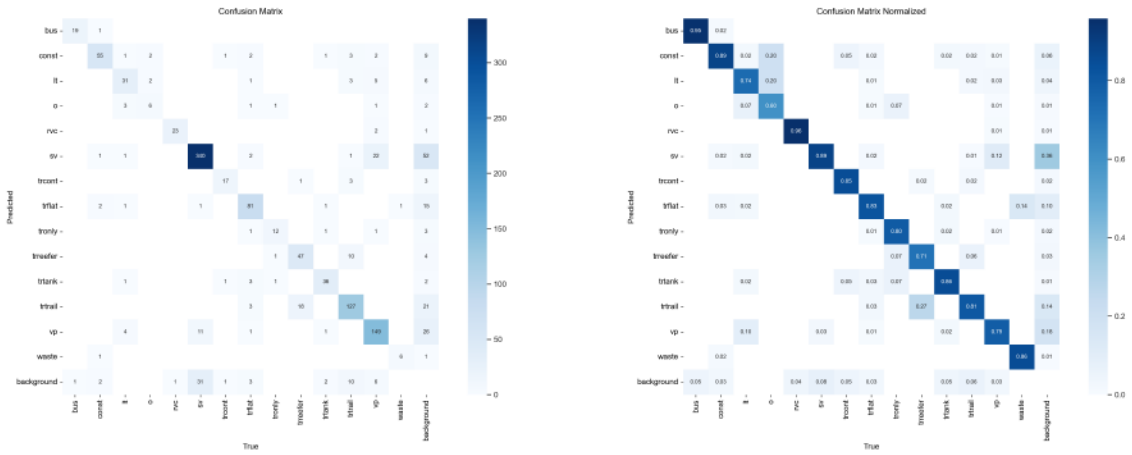


Figure 4.5: Quantitative results of YOLOV8x in our baseline dataset using non-normalized and normalized confusion matrices.

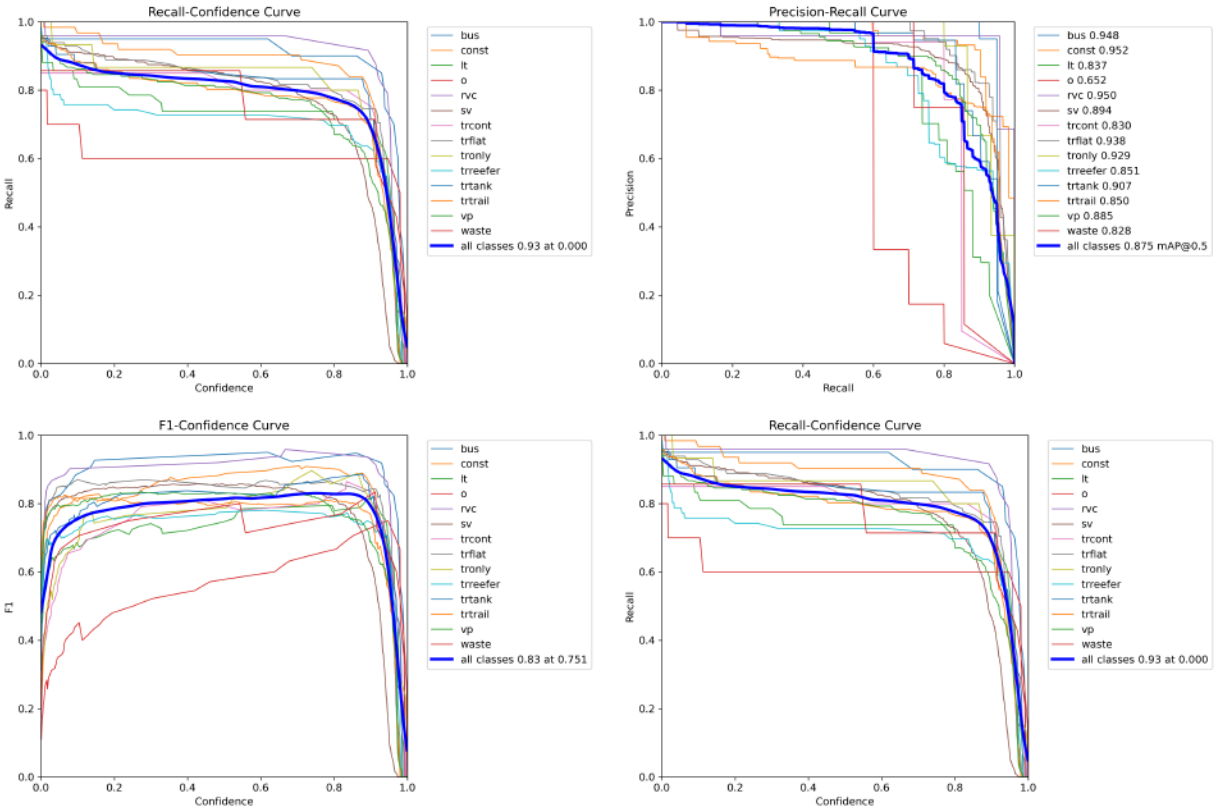


Figure 4.6: Quantitative results of YOLOV8x in our baseline dataset in terms of Recall-confidence Curve, Precision-recall Curve, F1-confidence Curve, and Recall-confidence Curve.

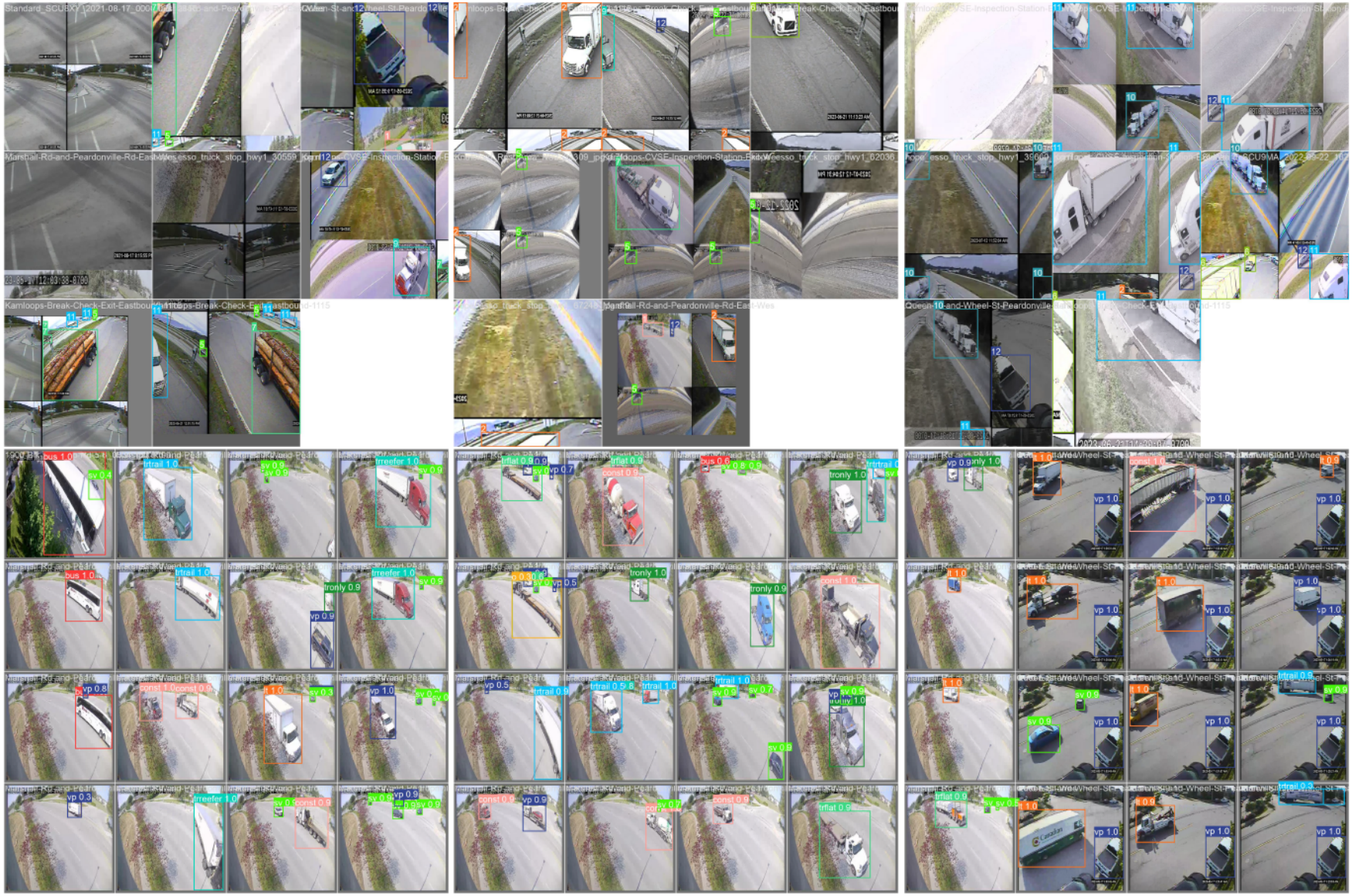


Figure 4.7: Qualitative results of YOLOV8x in our baseline dataset. Top: training batches. Bottom: validation batches with predictions.

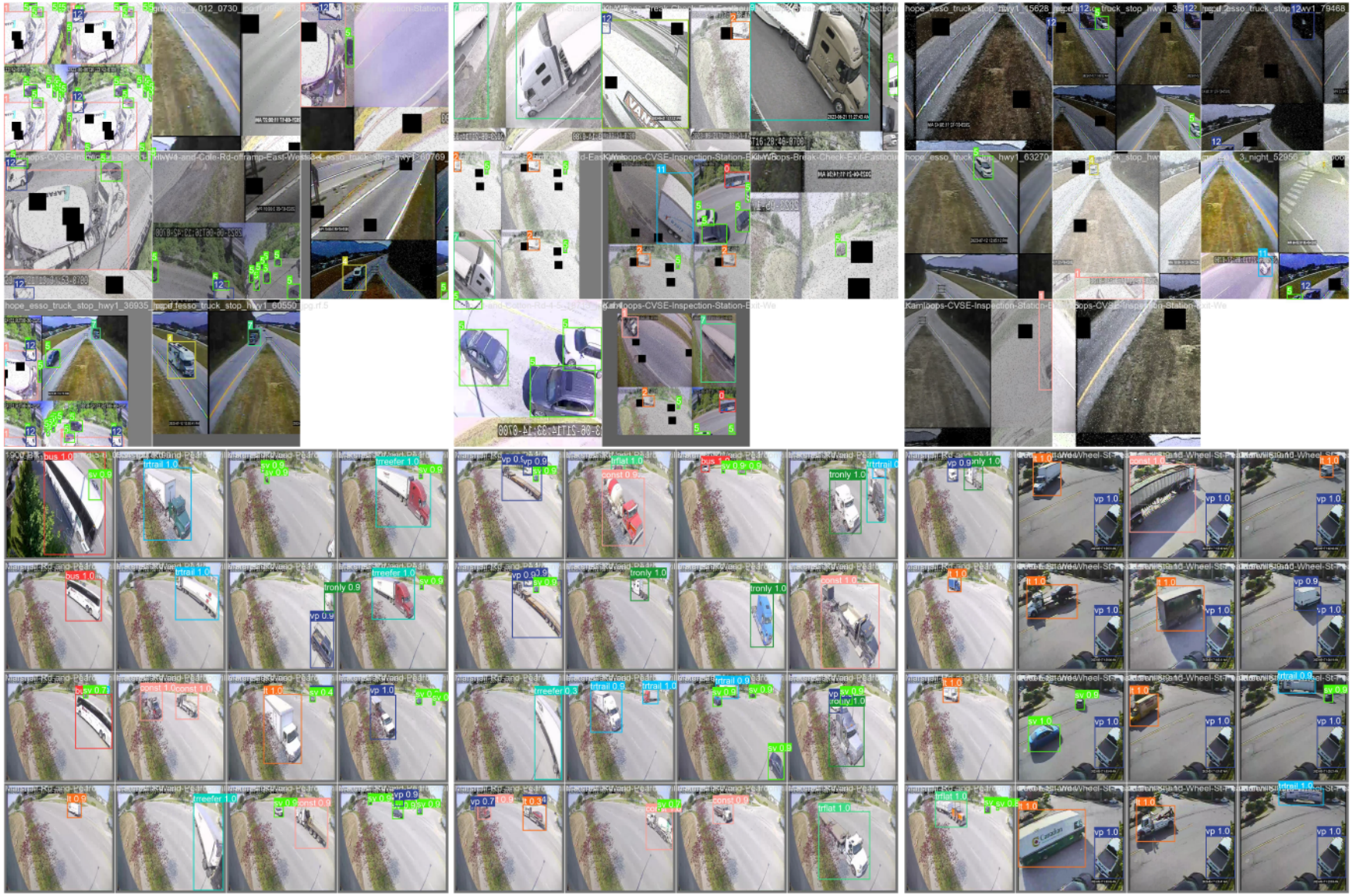


Figure 4.8: Qualitative results of YOLOV8x in the augmented dataset. Top: training batches. Bottom: validation batches with predictions.

4.5.3 Two-stage object detection results

Object detection for the two-classes dataset

On the experimental validation augmented dataset, YOLOv8x outperformed all other versions in terms of both mAP@0.5 and mAP@0.5-0.95, 0.968, and 0.855 respectively. However, a major constraint is its inference time of 27.6 for real-time traffic analysis.

Object Classification Results

Our dataset for object classification derived from our baseline dataset for object detection; we extracted all the bounding boxes' annotations as images and trained a YOLOv8l classifier. We performed comparative experiments when training with image inputs sized 128×128 and 192×192. The results show that the accuracy of top-1 and top-5 after fine-tuning with 128×128 are 0.83 and 0.98, respectively. However, the performance improves when using an image size of 192×192, with top-1 and top-5 accuracy of 0.84 and 0.99.

4.5.4 Truck analytics interface (UI)

We feel that a simple user interface (UI) can bridge the gap between complex machine learning models and non-technical users. It has higher accessibility and allows more people to learn about computer vision projects in a straightforward way. It simplifies the process of training models and classifying vehicles in general. Users do not need to understand the complexity of machine learning and can more easily utilize the project's functionality by simply interacting with the UI in order to overcome problems such as detecting and identifying complex vehicles, which greatly improves productivity. In the main display of our UI, the user can input a batch of videos into the system in a simple way and the results can be presented to the user in a way that is more visual, easier to understand, and easier to save and export. Users can click different buttons on the main page to further organize and analyze the data in a convenient manner. Furthermore, we also support users to select the appropriate models they need. In the model-selection page of our UI, we sort the different models according to their speed of recognizing objects, confidence level, etc. Users can use this information to effectively select the model they want to apply.

For better model training in the future, we added an active-learning button. This button allows for quick access to more labeled data of added value in a more time-efficient way. This greatly reduces the time and labor costs required to obtain labeled samples for future model training. In summary, the development of user-friendly

interfaces can greatly improve the accessibility and usability of machine learning-based vehicle recognition and classification projects, which not only expedites the adoption of such technologies but also reduces barriers in user training, enabling more people to take advantage of the application capabilities. As a result, a wider range of applications becomes attainable, leading to increased efficiency and productivity in the field of computer vision.

4.6 Conclusion

In this study, we have successfully developed an effective system for multi-class traffic analysis that exhibits a wide range of capabilities. This system excels in detecting, classifying, tracking, counting, timestamping, and estimating the direction of vehicles in diverse urban scenarios under varying conditions such as lighting and shadow. Our approach involves a carefully designed, two-stage process, starting with significant frame extraction, followed by the fusion of single-stage object detection with conventional algorithms for object post-processing and data analysis. Our comprehensive experimentation has yielded compelling results, showcasing the effectiveness and robustness of our system in vehicle analysis. We achieved high mean Average Precision (mAP) scores and recall rates when tested against our dataset. Moreover, our system is designed with adaptability in mind, as it incorporates a continuous learning loop. This adaptive approach ensures that our models evolve and improve over time as they receive continuous input data. In conclusion, our developed system not only excels in its current performance but also holds the promise of continuous enhancement, making it a valuable asset in the domain of traffic analysis and management.

Figure 4.9: Rows 1-3: Multi-class object detection and tracking examples using YOLOV8l and BoT-SORT tracker (one-stage approach). Rows 4-11: Object classification on predicted bounding boxes (two-stage approach).

Chapter 5

LiDAR feature extraction for WAM

5.1 Introduction

Urban city development depends on accurate collection of geomatics data with sufficient precision, resolution, and extent appropriate for engineering design. The term "geomatics" is defined by the International Organization for Standardization TC211 series as the tasks of gathering, transferring, storing, analyzing, processing, and presenting geographic information. It encompasses multiple disciplines such as geodesy, hydrography, mapping, navigation, photogrammetry, remote sensing, geographic information systems, and surveying.

Currently, ground-based and aerial-based geomatics, such as terrain modeling and land surveying (LS), are active research topics resulting from the rapid expansion of LiDAR (Light Detection and Ranging) sensing technologies, which creates more opportunities for the geomatics disciplines, including automation. Companies across the globe are engaging to advance this remote sensing technology and making use of its capabilities to retrieve useful data from the earth's surface. LiDAR technology benefits many applications such as agriculture, astronomy, atmosphere, autonomous vehicles (AV), forestry, biology, green energy, geology and soil, mining, and transport. LiDAR devices apply light in the form of a pulsed laser to measure ranges projected over natural and man-made environments to acquire information about surfaces, e.g., terrain, vegetation, and physical objects. Moreover, the light pulses combined with imagery data recorded by an airborne or terrestrial optical camera system provide useful clues with high-resolution, colorful, and three-dimensional information about the shape of the target's surface.

LS requires measuring the relative positions of natural and man-made features,

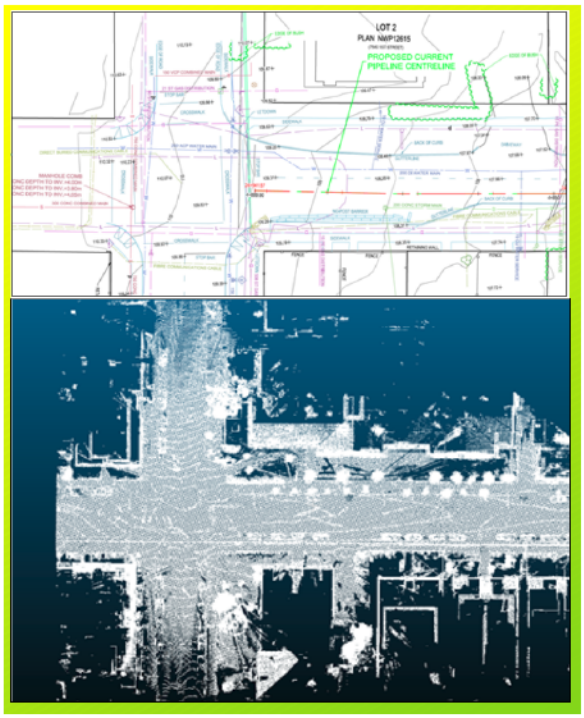


Figure 5.1: Top: CAD drawing generated from a conventional surveying routine, including accurate spatial information of urban assets. Bottom: point-cloud scene for the same location.

e.g., infrastructure, traffic signs, and utilities on or under the earth's surface. Traditional LS techniques involve manual operations to measure, calculate, and plan, in order to determine an accurate position of an urban object. For example, in a typical LS routine, a geomatics company sends a ground survey crew to the geographical area under study with accuracy devices such as a Global Navigation Satellite System (GNSS) receiver, prismatic compasses, clinometer, among others. The crew then localizes the target objects e.g., manholes, catch-basins, water-valves, and linear features that stand out in the urban environment, such as paint lines, stop bar paint lines, side walks, or curbs (top and bottom). The surveyor also determines heights and distances; identifies buildings, bridges, and roadways; determines areas and volumes; and draws plans at a predetermined scale. Although these conventional surveys and assessments can provide accurate geomatic data, they are predominantly manual tasks that require a high level of expertise to execute. Furthermore, the presentation of the geomatic information must be either by an engineering drawing as shown in Fig. 5.1, or by numerical values in the form of tables relying on expensive software tools. In view of the rapid expansion of cities and towns, the manual process becomes time-consuming, restricted, and limited to skilled labor. Therefore, automation with

the support of LS software is the current trend for increasing productivity. A preliminary, but necessary step is to automatically extract urban features from point-cloud data.

Table 5.1: Comparison of existing urban 3D LiDAR datasets.

Dataset	Year	Location	Classes	Imgs	Frames	Points	Scenes	LiDAR mode	Spatial Size	Task	Annotation
Semantic-3D [158]	2017	Europe	8	-	-	4B	30	TLS	-	Segmentation	point-wise
Oakland [159]	2009	Oakland, Pittsburgh	44	-	-	1.61M	17	MLS	1510 m	Segmentation	point-wise
IQmulus [160]	2015	Paris, France	22	-	-	300M*	1	MLS	210m	Segmentation	point-wise
Toronto-3D [161]	2020	Toronto, ON	8	-	-	78.3M	4	MLS	1000m	Segmentation	point-wise
Paris-Lille-3D [162]	2020	Paris - Lille	50	-	-	143.1M	3	MLS	1940m	Segmentation	point-wise
Street3D [163]	2020	-	5	-	-	290M	80	MLS	-	Segmentation	point-wise
ISPRS [164]	2012	-	9	-	-	1.2M	5	ALS	-	Segmentation	point-wise
DALES [165]	2020	Surrey, BC	8	-	-	505M	40	ALS	-	Segmentation	point-wise
LASDU [166]	2020	Heihe, China	6	-	-	3.12M	4	ALS	-	Segmentation	point-wise
DublinCity [167]	2019	Dublin	13	-	-	260M	12	ALS	-	Segmentation	point-wise
CAMPUS3D [168]	2020	Singapore	24	-	-	937.1M	1	UAV	-	Segmentation	point-wise
SensatUrban [169]	2020	UK	13	-	-	2847M	14	UAV	-	Segmentation	point-wise
Semantic-KITTI [170]	2019	-	28	-	23k/20k	4.5B	-	MLS	39.2km	Segmentation	point-wise
KITTI [171]	2012	Karlsruhe	8	15k	15k	-	-	MLS	-	Detection	2D/3D bbox
H3D [172]	2019	SF	8	83k	27k	-	-	MLS	-	Detection	2D/3D bbox
Argoverse [173]	2019	Miami, PT	15	490k	44k	-	-	MLS	-	Detection	2D/3D bbox
Lyft L5 [174]	2019	Palo Alto	9	323k	46k	-	-	MLS	-	Detection	2D/3D bbox
A*3D [175]	2019	SG	7	39k	39k	-	-	MLS	-	Detection	2D/3D bbox
Waymo Open [176]	2019	3x USA	4	1M	200k	-	-	MLS	-	Detection	2D/3D bbox
nuScenes [177]	2019	Boston, SG	23	1.4M	400k	-	-	MLS	-	Detection	2D/3D bbox
LiSurveying (Ours)	2021	Canada	54	-	-	2.45B	3	TLS	-	Detection & Segmentation	3D bbox/point-wise

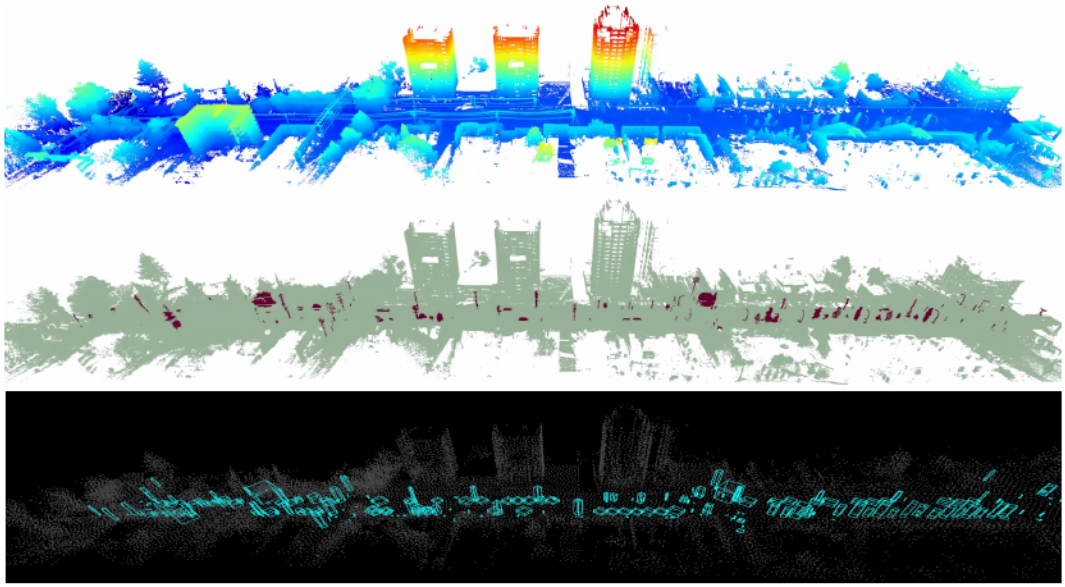


Figure 5.2: Top: New Westminster (NW) raw point-cloud scene. Middle: NW point-cloud with point-wise points. Bottom: NW point-cloud with 3D bounding boxe annotations.

5.1.1 LiDAR sensing modalities

Current 3D datasets in urban scenarios are mainly differentiated by their applications and gathering modalities. The existing datasets are divided into three main categories of LiDAR modes: airborne laser scanning (ALS), mobile laser scanning (MLS), and terrestrial laser scanning (TLS).

Algorithms designed to work with ALS differ considerably from those developed for MLS and TLS. One difference between terrestrial and mobile laser scanning devices is that TLS acquires data from the ground on a stationary platform or fixed location on the ground, while MLS data is acquired from a non-stationary platform, normally on the top of a vehicle. In contrast, ALS collects point-cloud data from the air with the sensor field of view pointing straight down at the ground (also known as "nadir"). Another important difference in these modalities is the point-cloud acquisition resolution. The point-cloud resolution dictates the information detail on the surfaces, which might vary depending on the application and device specifications. For surveying and engineering, high-resolution point-clouds are required to obtain accurate information about urban objects. These remote sensing modalities aim to capture point-clouds and use algorithms to automatically process the raw scanned data, and later produce a suitable output, e.g., 3D bounding boxes, semantic segmentation, or pose of the object.

Figure 5.3: Top: Google Street View image of 71 Merivale St New Westminster, British Columbia. Bottom: point-cloud for the same location with some examples of ground truth bounding boxes in our LiSurveying dataset.

5.1.2 Urban LiDAR datasets

The *KITTI* [171] is a pioneer multimodal dataset that includes point-clouds from a LiDAR sensor as well as front-facing stereo images and GPS/IMU data. This dataset with 8 classes is useful for AV applications using MLS point-clouds and front-facing stereo images. This dataset contains about 15k point-cloud frames with 200k 3D bounding boxes on 22 different scenes. However, due to the low resolution of points per frame, fine urban features are lost from the scene. *Waymo Open* [176] provides significantly more annotations, mostly due to the higher annotation frequency and system setup. Five LiDAR sensors and five high-resolution pinhole cameras were used for data collection. Like KITTI, this dataset enables tasks such as 3D object detection and tracking, but it only annotates four classes. *NuScenes* [177] is a recent dataset for AV with more annotations than KITTI and Waymo Open. Scenes include high traffic densities, potentially dangerous traffic situations, and situations that may be difficult for an AV. Despite this dataset including 23 classes, it contains only low density of points per frame and most of the classes are not related to LS tasks. Previous datasets allowed for the evaluation of a limited number of different classes for autonomous systems. Comparably, H3D [172], Argoverse [173], Lyft's L5 [174], and A*3D [175] are useful for the development of reliable self-driving vehicles, including trajectory predictions of agents such as pedestrians, vehicles, or bicyclist on urban roads. However, employing these datasets for surveying and engineering, is impractical due to the lack of classes relevant to land surveying applications.

Other datasets contain dense point-cloud scenes but are created for semantic segmentation. For example, *Semantic-3d* [158] is a TLS dataset that contains dense data (4 billion points). Different viewpoints captured using a static sensor are registered. However, this dataset contains only eight classes. *Toronto-3D* [161] was acquired with a vehicle-mounted MLS system on Avenue Road in Toronto, Canada. It poses challenges due to noise and the variations of point densities at different distances. It has eight classes (+1 unclassified points), including labels such as road, natural, building, and fence. *Paris-Lille-3D*[162] is one of the largest (2 km) and popular MLS point-cloud dataset for semantic segmentation. It contains 50 classes where nine classes from four difference scenes are used as benchmark. Oakland dataset [159] introduced 17 scenes with 44 urban classes similar to [162]. Despite these two datasets containing many surface utilities (pole, lamp, meter, traffic signs) which are essential for land surveying,they both lack linear features and flat objects. There are other datasets for 3D semantic segmentation such as IQmulus [160], Street3D [163], ISPRS [164], DALES [165], LASDU [166], DublinCity [167], CAMPUS3D [168], Sen-

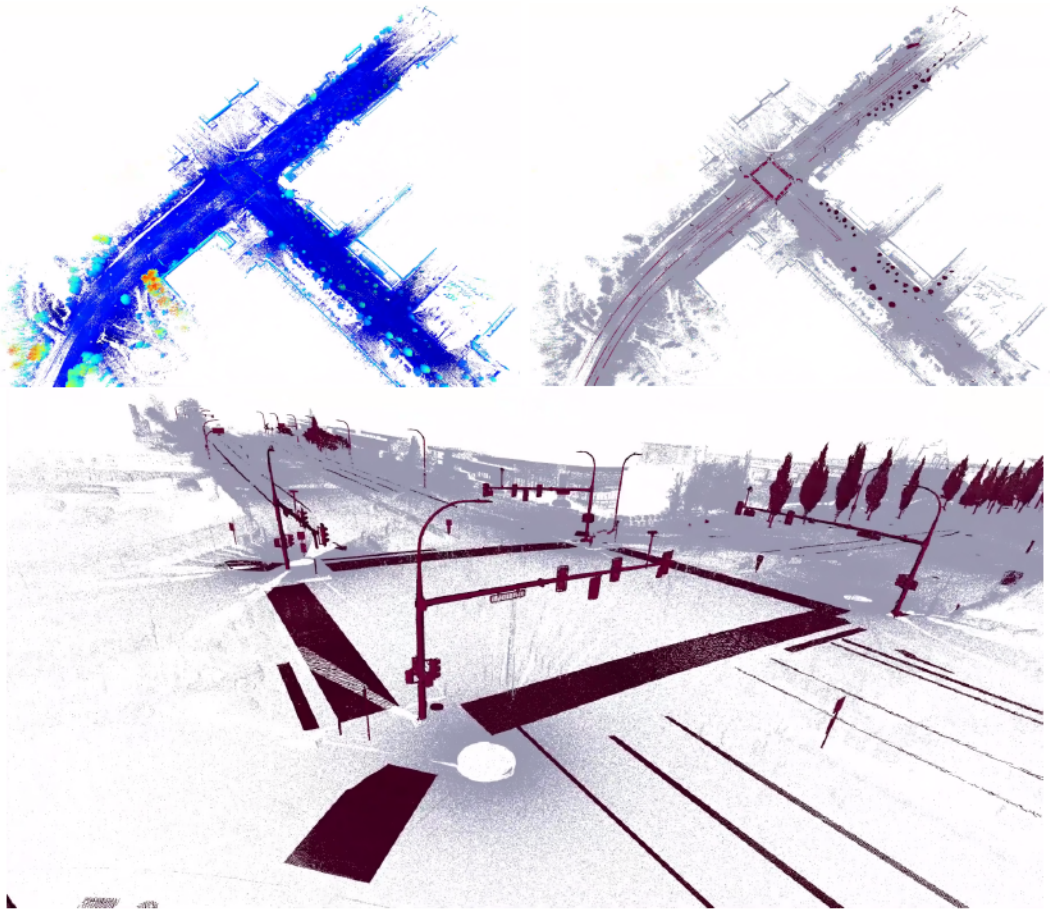


Figure 5.4: Top: MW raw point-cloud scene (left side) and MW with point-wise annotation from top view (right side). Bottom: MW point-wise annotation from side view.

satUrban [169], and Semantic-KITTY [170], but the number of object classes is very limited for our application domain. In conclusion, these datasets need more object classes, as listed in Table 5.1, to perform a complete surveying task on urban scenes. Therefore, there is a need to introduce a suitable dataset for surveying and engineering in order to train and validate appropriate detection and classification methods. We introduce the LiSurveying dataset, which contains three dense point-clouds of urban scenes located in Vancouver, Canada. While other datasets contain thousands of point-cloud frames, LiSurveying needs only three high-resolution point-clouds, one for each scene captured by multiple scanners during acquisition. The unique characteristics of our LiSurveying dataset include: a diverse range of object sizes, shapes, and details, and a large number of uncommon urban object classes valuable for land surveying. An additional aspect that distinguishes our dataset from existing urban datasets is that our ground truth data extraction is based not only on 3D bounding

boxes and point-wise annotations, but also on a validation of the annotation with object information in the physical world. This validation is conducted by comparing the annotated points with the CAD engineering drawings provided by domain experts, which represent the real, precise, and reliable spatial locations of the point features and linear features in the ground coordinates system. Ground coordinates are measurements that are taken on the actual surface of the earth. In this way, the data obtained by algorithms can be analyzed and compared more rigorously with real-world assets. A challenging objective is to be able to process the point-cloud in an outdoor scenario at different scales so that points and lines of interest can be extracted automatically to generate visualization, as shown in Fig. 5.1. In the land surveying application domain, algorithm precision is more important than time performance. However, generating a huge volume of data for numerous classes of objects in dense point-clouds is still a challenge in the research community.

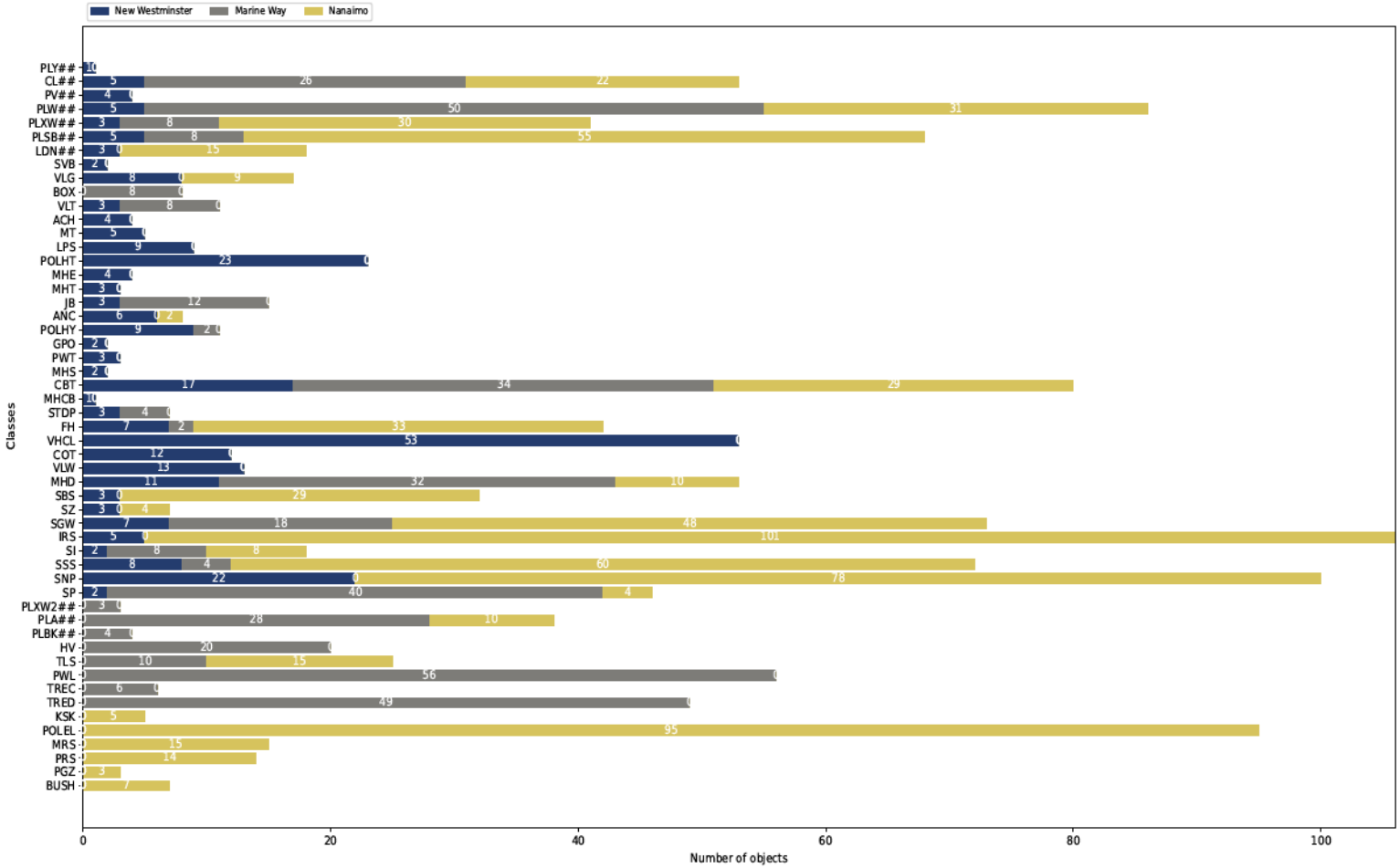


Figure 5.5: Number of objects per class on NW, MW, and NN.

5.1.3 Challenges of constructing LiSurveying

There are classes that are similar in terms of shape and size. In order to avoid incorrect labeling, the annotation of our dataset requires high precision, and thus is time-consuming. So far, we have performed annotation of around 5.3 million out of 2.45 billion points. We prioritized point-cloud objects not already covered or insufficiently present in current datasets, and we selected urban assets that are important for land surveying and engineering applications. Since big objects or large surfaces, such as buildings, bridges, pavement, or natural low/high vegetation terrain, are not our application focus, we left out those points from the 2.45 billion raw data. Labeling large objects can easily increase the total number of labeled points but is not crucial for our application domain.

In a point-cloud scene, we annotate urban objects, e.g., lamp stand, electrical transformer, or a valve on the ground. From a surveying perspective, it is important for a surveyor to localize the spatial locations of a variety of urban assets, such as gulleys, parapets, pipes, fuel drains, and small features. The unannotated points can be labeled as an "unknown" class for scene segmentation and modelling purposes, but they are not needed for object detection or recognition in our application.

While we provide a benchmark dataset for research and development (R & D), we leave the data imbalance issue to be handled at the R & D stage. For example, objects like trees and vegetation, as well as vehicles and pedestrians, are commonly found in LiDAR captured, outdoor point-clouds. On the other hand, urban utility objects like those we collected are typically uncommon and scarce. When a learning-based process requires a combination of both types of assets, either the overfitting samples are downsized, or data augmentation techniques are adopted to address the data imbalance issue. This is a separate research problem for machine learning model training and is discussed in Section 5.2.4.

Existing datasets lack the uncommon object classes that we have contributed. Different approaches, e.g., computer vision, machine learning, and deep learning, can be adopted by researchers based on the application requirements, volume of point-cloud available, and the object features to be detected. Regardless of which approach is selected, our dataset can be useful as it contains uncommon urban assets not covered, or inadequately covered, in existing datasets.

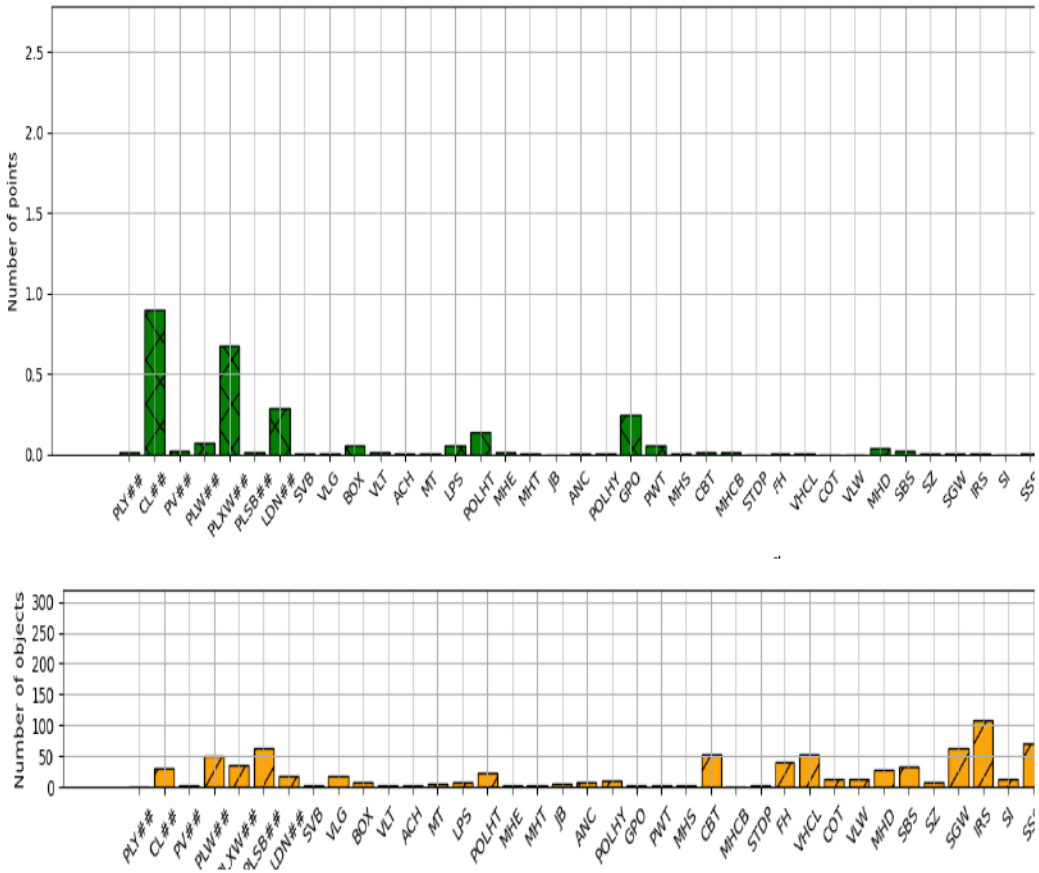


Figure 5.6: Top row: number of points per class overall in LiSurveying dataset. Bottom row: number of objects per class in LiSurveying dataset.

5.2 LiSurveying dataset description

This dataset was acquired with an ultra high-speed time-of-flight device enhanced by Waveform Digitizing (WFD) technology Leica ScanStation P20 sensor. Point-clouds were acquired in multiple locations using 360 degrees horizontal and 270 degrees vertical. The sensor's scan wavelength was 808 nm (invisible) / 658 (visible) with a range of 120 m; there was 18% reflectivity (minimum range 0.4 m) and a scan time and resolution of 12.5mm @ 10m. The dataset consists of three dense point-cloud scenes in British Columbia, Canada defined as follows: New Westminster (NW), Marine Way (MW), and Nanaimo (NN). We worked on these scenes due to their diversity across locations in terms of surface utilities, roadworks, catch-basins, manholes, street intersections, different vegetations, multiple buildings, pavement markings, hydro and electricity utilities, and a variety of traffic signs. NW covers the area of the city of New Westminster (see Fig. 5.2 and Fig. 5.3), specifically Agnes St. and FS Elliot St., Westminster (1.901 km). NW contains a high number of urban points and linear features along Agnes St. making this point-cloud scene very challenging to process but reliable for automatic urban surveying. It includes a variety of objects that experts in LS target to locate, e.g., valves, posts, culverts, or sidewalks. MW point-cloud is larger than NW, covering a selected area of Burnaby, specifically Marine Way St. (see Fig. 5.4). MW includes many curves and irregular terrains that make it a very complex scene. The urban objects in MW are more dispersed than those in NW. In contrast, the NN scene is captured over a large area (3.460 km) over Nanaimo St. This is the largest scene, including more instances than NW and MW. A challenge of NN is that there are many occlusions due to terrain variations and dense vegetation along the streets. Each scene is divided into multiple subsets for efficient processing. NW, MW and NN are divided into 18, 7, and 30 subsets respectively. We distinguished each subset with a unique number and color for reference during the annotation process.

5.2.1 Point cloud annotation

Object annotation is a labor-intensive task across disciplines, whether it is medical data or land surveying data. The object categories are defined by geomatics experts for applications relating to surveying and engineering design. In geodatabases, physical objects, surfaces, or terrains are classified mainly into seven groups: points, lines, polygons, annotations, dimensions, multipoints, and multipatches. Point features refer to permanent physical objects (typically man-made objects) that are located along the street, curb, or road that cannot be represented as lines or polygons, e.g., a fire hydrant or catch-basin. In contrast, lines or linear features generally refer to flat and

Figure 5.7: Top: A subset of NW scene with point-wise labels. Bottom: A subset of MW scene with point-wise labels

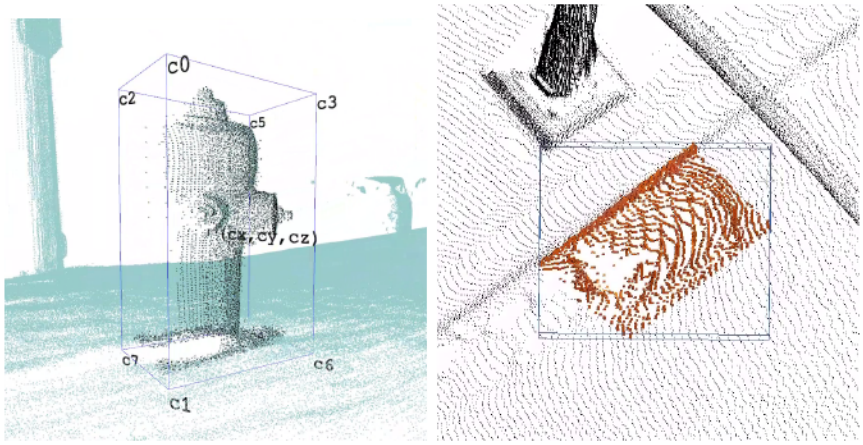


Figure 5.8: 3D bounding box annotation examples for hydrant and catch-basin objects.

long geographic objects or surfaces such as street center-lines or traffic paint lines. This means that linear features are very narrow compared to areas or volumes. The polygon category represents the structure and spatial location of homogeneous feature types, e.g., parcels or land-use zones. Annotation generally refers to adding text data to provide the description, information, layer, or property of another feature. The dimension category represents a type of annotation that contains specific lengths or distances. Multipoints and multipatches, respectively, are used to manage arrays of very large point collections or represent the outer surface of features that occupy a discrete area or volume in 3D space.

In LiSurveying, there are 54 categories containing point features and linear features. Categories are segmented and annotated manually with 3D bounding boxes, as shown in Fig 5.7 and Fig 5.8, respectively. Each bounding box contains the following parameters: center coordinates (cx, cy, cz), height, width, and length of box (h, w, l), box corners coordinates (m-by-n matrix, with m=8 and n=3), maximum bounds for geometry coordinates $(maxx, maxy, maxz)$, minimum bounds for geometry coordinates $(minx, miny, minz)$, and unique ID. It also contains object coordinate points (obj_x, obj_y, obj_z), rgb color for each point, and intensity value of each point. The object classes are divided into 7 main types:

- Roadworks: These include different paint lines on the road, intersections, and traffic paint lines.
- Surface utilities: These are different small surface utilities such meters, access hatches, boxes, and vaults.
- Hydro/tel: These are electricity and telephone utilities such as utility poles, different types of lamps, and anchors.
- Sto/san/water/misc: This type includes a variety of flat utilities like manhole covers, and catch-basins.
- Signs: This includes all traffic signs that can be found in the streets, e.g., bus stops, school zones, information signs, etc.
- Vegetation: This type includes trees with trunks in different sizes and shapes.
- Transportation: This covers all vehicles on the scene.

A summary of class labels and description of labels is shown in Table 5.2.

Table 5.2: Class labels and description in LiSurveying

Class	Description	Category	Class	Description	Category
PLY##	Paintline-Yellow	Roadworks	MHS	Manhole-San	sto/san/water/misc
CL##	Center-line	Roadworks	CBT	Catchbasin (Top Inlet)	sto/san/water/misc
PV##	Pavement	Roadworks	MHCB	Catchbasin Manhole	sto/san/water/misc
PLW##	Paintline-White	Roadworks	STDP	Standpipe	sto/san/water/misc
PLXW##	Paintline-Xwalk	Roadworks	FH	Fire Hydrant	sto/san/water/misc
PLSB##	Paintline-Stop bar	Roadworks	COT	CleanOut	sto/san/water/misc
LDN##	Letdown	Roadworks	VLW	Watermain Valve	sto/san/water/misc
PLXW2##	patinted xwalk	Roadworks	MHD	Manhole-Sto	sto/san/water/misc
PLA##	Paintline Arrow	Roadworks	SBS	Bus Stop	Signs
PLBK##	patiend bike	Roadworks	SZ	School Zone	Signs
SVB	Service Box	Surface utilities	SGW	Warning Sign	Signs
VLG	Gas Valve	Surface utilities	IRS	Information	Signs
BOX	Box	Surface utilities	SI	Speed Indicator	Signs
VLT	Vault	Surface utilities	SSS	Stop Sign	Signs
ACH	Access Hatch	Surface utilities	SNP	No Parking	Signs
MT	Meter	Surface utilities	SP	Sign Post	Signs
KSK	Kiosk	Surface utilities	MRS	Mandatory	Signs
LPS	Lamp Standard	Hydro/Tel	PRS	Prohibitory	Signs
POLHT	Hydro/Tel Pole	Hydro/Tel	PGZ	Play Ground Zone	Signs
MHE	Manhole-Electrical	Hydro/Tel	BUSH	Bush	Vegetation
MHT	Manhole-Telephone	Hydro/Tel	TREC	Tree-Coniferous	Vegetation
JB	Junction Box	Hydro/Tel	TRED	Tree-Deciduous	Vegetation
POLEL	Power Pole L	Hydro/Tel	TR	Trees	Vegetation
ANC	Guy Anchor	Hydro/Tel	VHCL	Vehicles	Transportation
POLHY	Hydro Pole	Hydro/Tel	GPO	Guy Pole	Hydro/Tel
PWT	Pole with transformer	Hydro/Tel	HV	Hydro Marker	Hydro/Tel
TLS	Traffic Light Standard	Hydro/Tel	PWL	Pole With Light	Hydro/Tel

5.2.2 Source files

Each scene of the dataset is managed in a separate folder, and each subset point-cloud is saved as a .pts file. For NW, each .pts file includes x, y, z (float) values, RGB values (0-255), and intensity (uint8). In MW and NN, each .pts file includes x, y, z (float) values and intensity (uint8). Depending on the scene, additional source files are included, as shown in Table 5.3. These are ALS data (.pts format), orthoimage (.tiff), LS CAD drawing (.dwg), demographic location (.kmz), Digital surface model (.xml), and Codelist (.xlsx).

Table 5.3: Summary comparison of our LiSurveying scenes

Scene	Points	TLS	TLS Intensity	TLS RGB color	ALS	Orthoimage	CAD drawing	Longitudinal Location	Digital Surface	Codelist
New Westminster	226,444,598	✓	✓	✓	✓	✓	✓	✓	✓	✓
Marine Way	622,144,987	✓	✓	✗	✗	✗	✓	✓	✓	✓
Nanaimo	1,602,623,087	✓	✓	✗	✗	✗	✓	✓	✓	✓

5.2.3 Dataset statistics

This section describes the statistics on the LiSurveying dataset generally, as well as individual aspects of them. The comprehensive number of points is 2.45 billion, with around 5.3 millions have been labeled. Trees and bushes are the most prominent categories due to their frequent appearance on the different scenes. These vegetation categories represent around 3 million points. These are followed by tall objects such as Guy Poles "GPO," Pole with transformer "PWT," and Power pole objects "POLEL," with 245k, 213k, and 210k points, respectively. Information Road Sign objects are very frequently found in the different scenes, representing 90k points, followed by standard lamps with 88k points. The number of objects per class and distribution of points per object class in the LiSurveying dataset are shown in Fig. 5.5 and Fig. 5.6, respectively.

In the NW scene, the longest category in terms of points is located at "PLW;" however, it should be noted that this category corresponds to the type of roadworks which are quite extensive throughout the scene for being linear features. "GPO" and "PWT" are the categories of physical objects with the highest number of points in this scene, with both adding up to around 450k points. These are followed by the "LPS," "POLHT," "POLHY," and "TR" classes, which are represented by a number of points ranging from 60k to 90k. The categories "VLT", "VLG", and "ANC" have the least points with less than 10k. The latter in real-life are small objects located along roads or sidewalks, and therefore the number of points in each object depends on both the parameters of the acquisition system and the structure of the object. Note that certain objects might not exist in the NW dataset but exist in the MW or NN dataset. In the MW scene, the "TLS" class stands out, as there is a high occurrence of vegetation in this location, leading to a high number of points close to 800k. Unlike NW, in MW some classes such as "VLG," "ACH," "MT," or "COT" are not found. However, classes related to traffic signs exhibit similar occurrences in both scenes. In the NN scene, the class with the highest number of points is "TR," which contains different types of tall trees, while the classes with the fewest points are "ANC", "SZ" and "FH". Fig. 5.5 shows a comparison of the number of objects in each class. Overall "TR" is the class with the highest number of objects with around 500 samples. It should be noted that this class is not included in Fig. 5.5 so that the other classes can be more clearly visualized. "IRS" has the second-highest number of object (106 samples), followed by "SNP" and "POLEL" with 100 and 95 objects, respectively. The class with the lowest number of objects is "MHCB" with 1 sample. Classes that have a low number of objects are not discarded, because more samples

can be obtained from even a single object with high point density and quality by using data augmentation techniques.

Variation of point density and shape

Unlike other datasets such as *KITTI* [171], *Paris-Lille-3D* [162] or *Toronto-3D* [161], LiSurveying has a greater variation in object shapes, dimensions, and point densities. The variation of point density stands out due to the complete configuration in 360-degree scanning in terrestrial static mode, in which multiple scans of the same scene are acquired and subsequently registered. Depending on the longitudinal and latitudinal positions of the sensor with respect to the scene or streets, certain objects reveal a greater number of points than other objects that are farther away from the source of acquisition. The scans were performed mostly at the corners of intersections and at spots between the edge of the road and the sidewalk, because most of the urban objects of interest are on the sidewalks. There are also variations in the regularity of points of some objects. This is because the objects far from the sensor were not scanned from all the surrounding angles. Therefore, some objects are partially scanned and others are completely scanned. Various shapes and dimensions of the objects are shown in Fig. 5.9. The classes called "tall objects" include poles, trees, lamps, kiosks, and fire hydrants. These have dimensions that range from 0.5 m to 14 m in height, 0.5 m to 2 m in width and length. Classes with flat objects are those embedded in the ground, such as manholes, catch-basins, and valves. These classes of flat objects can have circular, square, rectangular, or irregular shapes, but the dimensions are limited in the vertical axis with respect to the ground, and therefore they are called "flat objects."

5.2.4 Dataset partition

LiSurveying includes object classes with a variety of shapes including flat and non-flat, uniform and non-uniform, and big and small for classification tasks as shown in Fig. 5.10. We derived seven subsets from the LiSurveying dataset, as shown in Table 5.4. These were used in the experiments for training different models. The subsets are: Flat-9, Flat-10*, Tall-16, Tall-14*, Tall-25*, Merged-39, and Merged-39*. The Tall datasets normally include tall objects that have a dimension higher than 30 cm relative to the ground surface, e.g., fire hydrants, trees, poles, traffic signs with poles, and standard lamps. In contrast, the Flat datasets mainly contain objects that are parallel to the ground surface, e.g., catch-basins, manholes, valves, and flat boxes. These objects typically do not vary along the vertical axis. Note that the symbol *

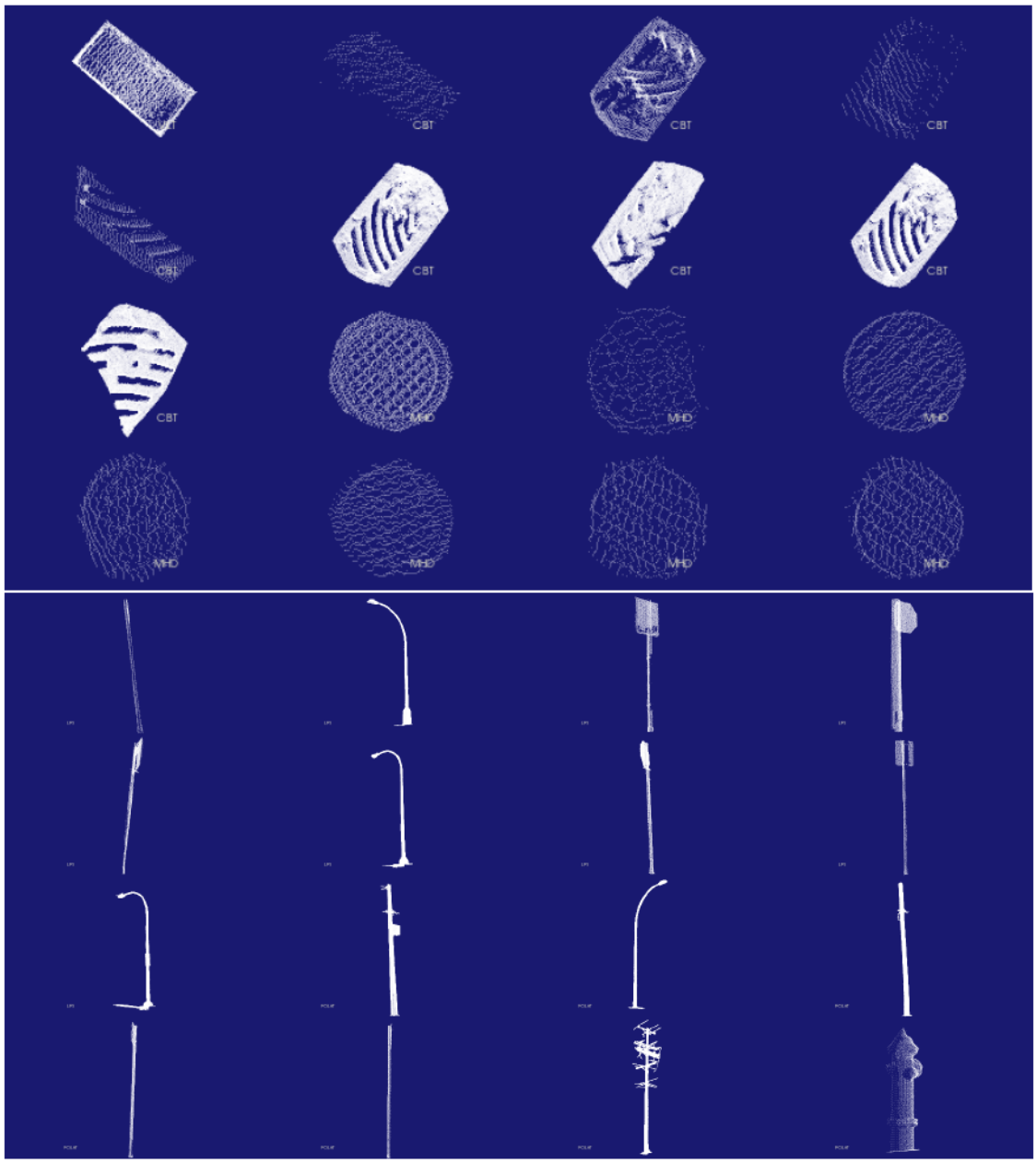


Figure 5.9: Comparison of flat objects (Top) vs tall objects (Bottom) in LiSurveying.

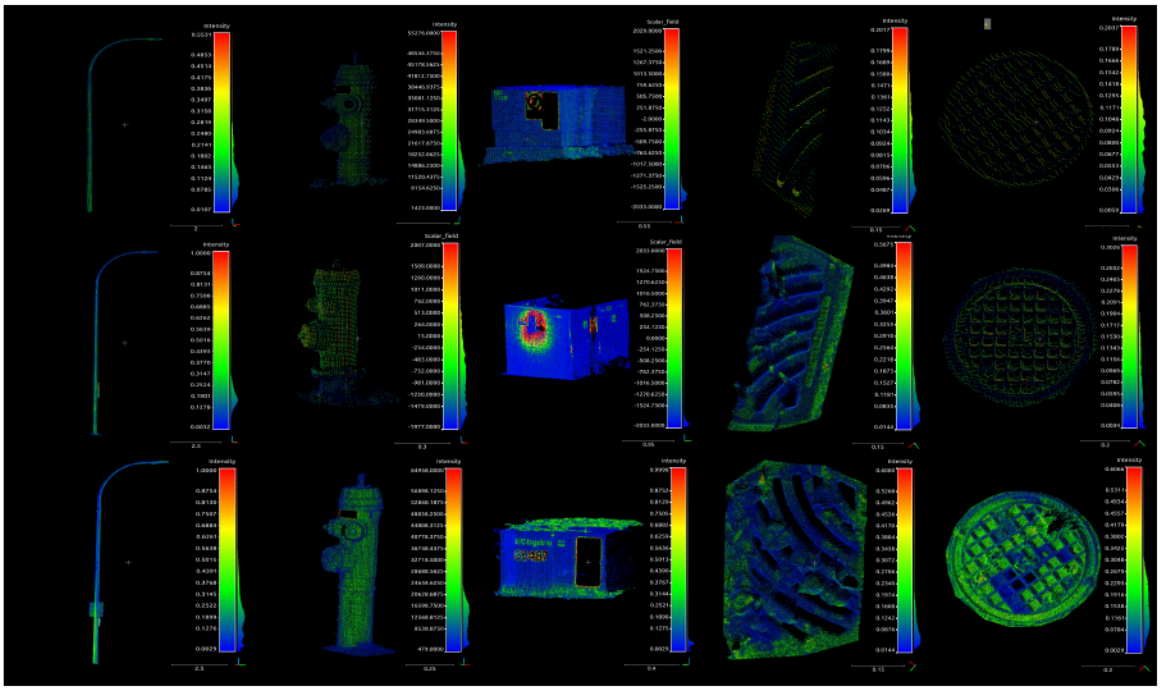


Figure 5.10: Point density comparison of three different objects for five classes. Objects are sorted from lowest to highest number of points (top to bottom). Left column: PLW objects with number of points: 6099, 153394, and 409278. Second column: FH objects: 2923, 3676, and 21134. Third column: Box objects with number of points equal to 13843, 56000, and 96459. Fourth column: CBT objects with 1541, 22741, and 37032 points. Right column: MHD objects with 2270, 5717 and 59755 points.

on specific datasets means that data augmentation was performed.

Data augmentation

Despite our LiSurveying dataset containing high-resolution samples per class, the number of samples in each class is not the same. Data augmentation is a common method used to address data imbalance. This solution aims to avoid overfitting, alleviate class imbalance problems, and achieve better generalization when training learning-based models. Therefore, we adopt this technique and in our data augmentation procedure, each point-cloud sample within its class has N points denoted by $\boldsymbol{S} = \{\boldsymbol{s}_i \in \mathbb{R}^3|_{i=1,\dots,N}\}$, and the N points for each sample are passed through our data augmentation pipeline's operations to generate more data samples. There are five main operations including random rotation R_θ, random translation T_r, random down-sampling, random down-sampling with random rotation, and jitter. The process starts by randomly rotating the interior points of $\boldsymbol{S}$ with a matrix R_θ following a uniform distribution $\Delta_\theta \in [0, 2\pi]$ along the z-axis (up-axis). We also apply ran-

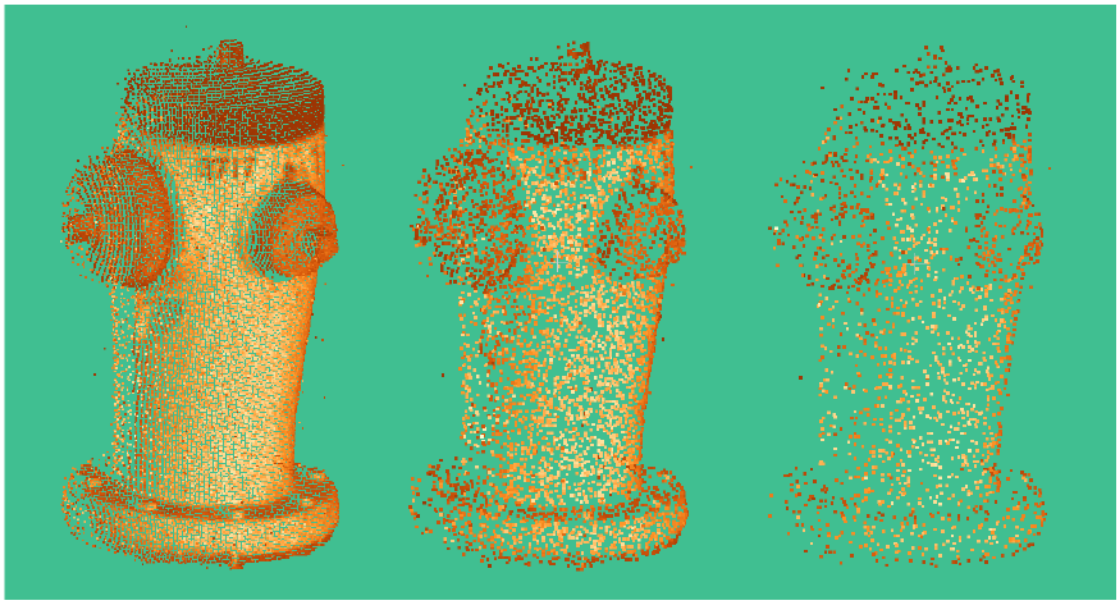


Figure 5.11: Example of a hydrant asset with 20895 points from New Westminster scene before (left) and after random downsampling to 4096 points (middle), and 2048 points (right).

dom translation $(\Delta_x, \Delta_y, \Delta_z)$ along the x-axis and y-axis. Then, we randomly down-sample each original point-cloud to 2,048 points. We also randomly down-sample each rotated-translated version and generate combinations of these versions. Down-sampling helps reduce memory consumption and efficiently reduces the size of our samples to as little as 1k, 10k, or more than 100k points depending on the class type, as shown in Fig. 5.11. Random jitter is also applied to the individual points, followed by random down-sampling. In our experiments, data augmentation was done systematically in all classes except for the TR (Trees/vegetation) class, which already had the desired sample size, which is 375 samples per class. For point-cloud classification tasks, we performed data augmentation in local coordinates. Different operations were applied to the points inside the 3D ground truth boxes.

5.3 Method Overview

5.3.1 Hand-crafted features

Since our application focus is land surveying and site analysis targeting specific urban assets, the selection of hand-crafted features is also based on the shape characteristics of these specific urban assets, e.g., points and lines. These shape characteristics are

also used by other researchers [178]. Covariance features are very popular for the extraction of representative information on 3D data. We select a set of local features using the covariance matrix of the points. The selection is based on object type: tall or flat. In this study, the covariance matrix (3 x 3) is calculated for each point-cloud object. Then, the resulting matrix is factorized into a canonical form using eigen decomposition in order to obtain the three eigenvalues (τ_0, τ_1, τ_2), and the three corresponding eigenvectors (v_0, v_1, v_2). These eigenvalues are used to calculate geometric features such as L linearity, P planarity, S scattering, V verticality, C_c curvature change, Eg eigenentrophy, Omn omnivariance, and An anisotropy. For tall objects, most changes occur in the vertical axis but there are small changes on the horizontal plane. The range on the z axis is selected as the height feature; different tall objects can have different heights from the ground. For example, a pole is taller than a road sign pole, but a fire hydrant is smaller than the latter. The standard deviation is also included as a feature in each of the axes to measure the dispersion relative to the mean of the points. For shape representation, the characteristics of verticality, curvature, linearity, planarity, scattering, omnivariance, anisotropy and eigentrophy are used. On the other hand, different features are selected for flat objects. However, these vary mostly on the horizontal axes (x and y) relative to the ground surface. Thus, the ranges in the x and y axes, standard deviation of the points, eigen feature entropy of x extremes, and eigen feature of y extremes are measured. As observed in our experimental analysis, L, P, S, V , and C_c, e.g., Eigenentrophy, Omn omnivariance, and Anisotropy, describe very well the features of our application's urban assets listed in Table 5.2. For commonly seen objects, e.g., trees, vehicles, and city blocks, they may need other specific hand-crafted features. Some of the equations are described as follows:

$$L = \frac{\tau_0 - \tau_1}{\tau_0} \in [0, 1] \tag{5.1}$$

$$P = \frac{\tau_1 - \tau_2}{\tau_0} \in [0, 1] \tag{5.2}$$

$$S = \frac{\tau_2}{\tau_0} \in [0, 1] \tag{5.3}$$

$$Omn = \left(\prod \tau_i\right)^{\frac{1}{3}} \tag{5.4}$$

$$Eg = -\sum \tau_i \ln(\tau_i) \tag{5.5}$$

$$An = \frac{\tau_0 - \tau_2}{\tau_0} \tag{5.6}$$

where τ_0, τ_1, and τ_2 are the eigenvalues.

5.3.2 Machine learning classifiers

We used different individual and ensemble machine learning models in our experiments with the above-mentioned hand-crafted features. The algorithms Decision Trees, Logistic Regression, Random Forest, Multilayer Perceptron, Extra Tree(s), Gradient Boost, Support Vector Classification (SVC), Linear SVC, Passive Aggressive, Gaussian Naive Bayes, and Stochastic Gradient Descent (SGD) were used to perform classification tasks on LiSurveying.

Decision Tree algorithm is represented by a tree-like model of decisions along with possible predictions in a diagram.

Random Forest is represented by a combination of multiple classification decision trees, where each decision tree is generated using a random vector sampled independently from the input vector. Then, each tree provides a unit vote, and the majority votes determine the class.

Logistic Regression algorithm evaluates the input feature vector and provides a prediction of the probability that an event will occur. This algorithm maps each data point using a sigmoid function.

Gradient boost is known as an ensemble learner that builds an additive model in a forward stage-wise way based on a series of individual models. This method allows the optimization of arbitrary differentiable loss functions.

KNN method finds a group of p samples that are nearest to the unknown sample (*e.g.*, based on distance functions). From these p samples, the label (class) of the unknown sample is determined by calculating the average of the response variables (i.e., the class attributes of the p's nearest neighbours).

SVM method's basic concept is to build a hyperplane in every transformed feature area, splitting the full margin.

Gaussian Naive Bayes (NB) is a supervised learning algorithm which applies Bayes' theorem with the naive assumption of strong conditional independence between every pair of features. This method makes use of all the variables in the feature vector X and analyzes them independently as they are uniformly independent of each other. The Gaussian Naive Bayes method is the easiest to work with as it only needs to estimate the mean μ and the standard deviation σ from the training vector. This method assumes that features follow a normal distribution. Before training this model, feature normalization is applied to normalize the range of variables, reduce over-fitting, and accelerate the training time.

Passive Aggressive are maximum margin based algorithms, which have been mainly used in online learning.

SGD has been widely used for training linear classifiers in many applications. This algorithm demonstrates high performance in computation time with no loss in classification on large-scale machine learning problems. SGD minimizes an objective function by approximating the gradient for a randomly selected batch from the training data. In each iteration, SGD considers one sample and updates the weight vector using a time-dependent weighting factor until it converges.

MLP is a linear classifier algorithm similar to SGD. It predicts based on a linear function combining a set of weights with the input feature vector. It has shown high performance for large datasets and can be used in online learning. MLP classifiers are faster in training. The algorithm is not regularized or penalized and converges to an optimal solution for linearly separable patterns with no upper bound on the learning rate parameter. For a large amount of data classification, this type of learning is preferable.

Soft voting classifier is a popular meta-classifier for merging conceptually different machine learning algorithms for prediction through a majority voting strategy. Soft voting performs majority voting by using the argmax of the sum of the predicted probabilities of the class labels and generates an output label. In the proposed methodology, seven of the previously mentioned classifiers have been ensembled for training.

LDGCNN [179] method is an improved version of DGCNN [180]. The authors removed the transformation network and linked the hierarchical features learned from different layers in the DGCNN. This modification showed a better performance on classification and decreased the model size of the network.

PointNet method learns the spatial encoding of each point in the input point-cloud. Then, the extracted individual point features are aggregated to a global signature. A major limitation is this method's lack of ability to capture local context at different scales, which is later overcome by PointNet++.

PointNet++ method has shown acceptable results in applications that require 3D point-cloud object classification or segmentation. It leverages small neighborhood sets of points at multiple scales to extract local and high-level features. This process is robust and captures details embedded in the point-cloud data. The hierarchical structure consists of a set of abstraction (SA) levels. Each abstraction level has three key stages (sampling, grouping, and MLP layer) that extract multiple scales of local patterns and combine them logically according to local point densities. At each level, a set of points is processed and abstracted to output a set with fewer elements. Given the points within $\boldsymbol{S} = \{\boldsymbol{s}_i \in \mathbb{R}^3|_{i=1,\dots,K}\}$, these are subsampled using an iterative farthest point sampling method to obtain query points. At the grouping stage, a ball

query method is used to localize an upper limit k neighbors of each query within a radius r; each group corresponds to a local region. Each local region is abstracted by its centroid and local feature that encodes the centroid's neighborhood. The process is repeated until the features are obtained. Finally, the obtained features are transformed by a multi-layer perceptron (MLP) network, followed by three fully connected layers and a softmax function to obtain a probability class score.

5.4 Experimental results

Table 5.4: Datasets setup derived from LiSurveying

Dataset	Classes name
Tall-16	LPS, SGW, SNP, IRS, Combined sings (PGZ, SP, SZ, PRS, MRS, SI, SBS, SNP), SSS, FH, BUSH, KSK, TLS, BOX, SVB, ANC, TR, STDP, and POL C (GPO, POLHT, POLHY, PWL, and PWT)
Flat-9	CBT, JB, HV, VLT, ACH, MT COT, VL c (VLW and VLG), and MH c (MHCB, MHD, MHT, MHE and MHS)
Tall-25*	SSS, SGW, SI, SVB, POLHT, PWT, FH, GPO, LPS, IRS, SZ, TLS, POLEL, SBS, SNP, PRS, MRS, KSK, PGZ, BOX, STDP, SP, PWL, ANC, and POLHY
Tall-14*	SVB, PWT, FH, GPO, LPS, TLS, KSK, PGZ, BOX, STDP, PWL, ANC, SIGNS (SSS, SGW, SI, IRS, SZ, SBS, SNP, PRS, MRS, SP), and poles (POLHT, POLHY, and POLEL)
Flat-10*	VLW, VLG, MT, JB, COT, ACH, VLT, HV, CBT, and MH (MHCB, MHD, MHT, MHE, and MHS)
Merged-39	VLW, VLG, SSS, SGW, SI, SVB, MHS, MHT, POLHT, MT, PWT, FH, GPO, MHE, JB, LPS, IRS, MHCB, COT, SZ, ACH, TLS, POLEL, SBS, SNP, PRS, MRS, MHD, KSK, PGZ, BOX, STDP, SP, PWL, ANC, VLT, POLHY, HV, CBT
Merged-39*	Merged-39 dataset after augmentation

We performed a classification on the seven subsets of the LiSurveying dataset: "Flat-9, Flat-10*, Tall-16, Tall-14*, Tall-25*, Merged-39, and Merged-39*" with conventional classifiers with hand-crafted features, LDGCNN, PointNet, and PointNet++. The previous datasets were used to investigate the impacts of different algorithms on the classification results. For training and evaluation, we used an Intel Core i5-9400 CPU (2.90GHz x 6), a TB hard disk, and a NVIDIA TU117 GeForce GTX 1650 with 16 GB on Ubuntu 20.04.4 LTS.

Table 5.5: Results on the different setups of LiSurveying using different learning based approaches.

Dataset →	Tall-16					Flat-9					Tall-25 +aug	Tall-14 +aug	Flat-10 +aug	Merged-39	Merged-39 +aug
Feature Scaling →	None	Scaled	Min Max	Scaled PCA	Min Max PCA	None	Scaled	Min Max	Scaled PCA	Min Max PCA					
Model															
Decision Tree	0.7419	0.7298	0.7460	0.6855	0.7298	0.5536	0.5714	0.5536	0.5179	0.6786	0.5423	0.7732	0.7319		
Logistic Reg.	0.7379	0.6895	0.5806	0.7056	0.5605	0.4643	0.5893	0.4107	0.6607	0.5536	0.4983	0.7244	0.5765		
Random Forest	0.8185	**0.8226**	**0.8306**	0.8145	0.7984	0.7143	0.6964	0.6786	0.6786	0.6071	0.6421	0.8403	0.8476		
MLP	0.8105	0.7782	0.7944	0.7500	0.7823	0.6964	0.6786	0.6607	0.6964	**0.6964**	**0.7353**	0.9138	0.7851		
Extra trees	0.8185	0.8024	**0.8306**	0.7944	**0.7984**	**0.7500**	**0.7679**	**0.7679**	0.7321	**0.6964**	0.5847	0.8311	0.8184		
Extra tree	0.6371	0.6089	0.6371	0.5927	0.6734	0.5357	0.5536	0.5714	0.5357	0.5893					
Gradient boost	0.7742	0.7702	0.7742	0.7339	0.7218	0.6786	0.6607	0.6786	0.6786	0.6607	0.6311		0.8069		
KNN	0.7903	0.6774	0.6653	0.6492	0.7177	0.6071	0.6429	0.6964	0.6429	0.6786	0.5522	0.7833	0.6531		
SVC	0.5484	0.6815	0.6694	0.7016	0.6935	0.4821	0.6429	0.6429	0.6964	0.6964					
linear SVC	0.6976	0.6734	0.6210	0.6895	0.6290	0.4821	0.5357	0.5357	0.7143	0.6429					
Nearest cent.	0.5403	0.4919	0.4718	0.4919	0.5000	0.5357	0.4286	0.4643	0.5179	0.5714					
Passive A.	0.4960	0.5927	0.5645	0.6371	0.5282	0.3214	0.5536	0.5714	0.6250	0.5893					
Ridge	0.5726	0.5565	0.5645	0.5685	0.5685	0.3571	0.4821	0.4464	0.5714	0.5893					
SGD	0.5806	0.6815	0.6532	0.6815	0.5565	0.4286	0.3750	0.4464	0.6607	0.5179					
Gaussian	0.7984	0.6855	0.6290	0.6774	0.5605	0.5000	0.6071	0.3750	0.6607	0.4643					
Gaussian NB	0.7863	0.7863	0.7863	0.7581	0.7581	0.4821	0.4821	0.4821	0.5179	0.5179	0.3877				
Voting				**0.9602**					**0.9352**		0.6809	0.8917	**0.8522**		
Random Search				0.6521					0.6641		0.6411		0.7732		
Random Search L2				0.7396					0.7411		0.5041		0.5821		
Hyperband				0.9473					0.9138		0.6847	0.8548	0.8133		
Hyperband L2				0.7669					0.7236			0.7003	0.5481		
PointNet	0.4137					0.1721					0.1798		0.3759	0.2231	0.3501
PointNet++	**0.8376**					0.7358						**0.9858**			
LDGCNN	0.7633					0.6011									

5.4.1 Implementation details

For the baseline classifiers, the datasets for Tall objects contain 13 covariance features extracted from each sample along all the axes. Datasets including Flat objects also contain 13 covariance features but are only extracted from the x- and y- axes. All features are extracted on the fly and concatenated as a feature vector. Note that we used 70% for training and 30% for testing. Then, for each classifier, we performed a random search on the hyperparameters with four cross-validation strategies, and fine-tuned feature scaling with PCA. To evaluate baseline classifiers, we use precision, recall, and f1-score. We evaluate the different models by considering their respective confusion matrices on both non-augmented and augmented datasets. The PointNet architecture used in our experiments has two essential components: multi-layer perceptron (MLP) networks and Transformer Nets (T-Net). The model pipeline receives a point-cloud object at the input layer. The first T-Net then learns an affine transformation matrix by its own internal network. This initial T-Net transforms the input points into a canonical representation. Afterwards, feature extraction is performed by two convolutional layers with 32 channels. Consecutively, a second T-Net is used to

Table 5.6: Hyperparameters setup for PointNet++.

Hyperparameter	Values	Description
N_{points}	$\Delta[128 - 2048]$	Number of the training points
B_{size}	32	Batch size
$N_{classes}$	[9, 10, 14, 16, 25, 39]	Number of classes
L_R	0.001	Initial learning rate
$Decay_{rate}$	0.0001	Initial learning decay
$Epoches_{max}$	250	Number of training epoches
$Step_{size}$	20	StepLR step size
$Step_g$	0.7	StepLR gamma
Optimizer	Adam	Optimization algorithm with L2

perform affine transformation for alignment in feature space (N, 3), to obtain (Nx32) dimensional point features. Three convolutional layers with 32, 64, and 512 channels are applied to the previous features, respectively. The resulting Nx512 dimensional features are converted to dimensional global features by a max-pooling layer. Finally, the global feature vector is reduced layer by layer through MLP with nodes 256 and 128, and a dropout layer between each pair of fully connected layers. The classification occurs in the last layer with a softmax activation function. For training, we used an Adam optimizer with an initial learning rate of 0.001, and a decay learning rate of 0.9. The model is trained for 200 epochs with a mini-batch size of 32, and 2,048 input points per training sample. The PointNet++ classification network receives a point-cloud with spatial coordinates and face normals at the Set Abstraction (SA) stage. The SA stage consists of three SA layers. In the first SA layer, 512 sample points are selected from the input. Local regions are delimited by using 32 nearest neighbors of each point within a 0.2 radius. From these local regions, features are extracted using MLPs (64,64,128). Then, the extracted features are summarized for 512 regions with max pooling. Similarly, in the second SA layer, features are extracted from 128 different local regions with MLPs (128,128,128). The third SA layer receives the remaining points for feature extraction. After generating the local features from the first and second SA layers, the global features in the third SA layer are extracted using MLPs (256,512,1024) with max pooling summarization. The final 1024 dimensional global features are linked to the labeled classes through MLPs (1024, 512, 256) and a set of neurons, the size of which depends on the number of input classes. In all layers except the last one, ReLU, batch normalization, and 0.5 dropout are applied. The hyperparameters used in our experiments are shown in Table 5.6.

5.4.2 Quantitative results

In this section, we conduct a quantitative comparison of the classifiers using hand-crafted features. Seven subsets of the LiSurveying dataset were used in the experimental stage: Flat-9, Flat-10*, Tall-16, Tall-14*, Tall-25*, Merged-39, and Merged-39*. This diversity of configurations in the seven subsets allows us to better validate the performance of the algorithms based on the variety of classes in LiSurveying. To compare the effectiveness of the models, we use Precision (P), Recall (R) or sensitivity, Specificity (S), and F1 Score. The experimental results are showed in Table 5.5. We also provide confusion matrices, which are plotted to intuitively show the strengths and weaknesses of the methods. Finally, we compare the performance of conventional classifiers with PointNet and PointNet++. We also evaluate the robustness of PointNet++ method with different input point size parameters during training, and at different SA layers.

5.4.3 Classification on Tall-16

Regarding our Tall-16 dataset, the ensemble soft voting classifier achieved the maximum accuracy value of 96.02% as compared to other classifiers using scaled PCA for feature scaling. The Hyperband algorithm showed the second-best performance value of 94.73%. Random Forest and Extra Trees both showed the highest accuracy value of 83.06% when applying min-max normalization.

Extra Trees showed the lowest accuracy value of 79.84% for all feature scaling configurations. We noticed that the algorithms Nearest Centroid, Extra Tree, Passive Aggressive, Ridge, SGD, and SVC showed low performance of 54.03%, 63.71%, 49.60%, 57.26%, 58.06%, and 54.84%, respectively. Even after data normalization, these algorithms did not improve in performance. Note that other classifiers such as Decision Tree, Logistic Regression, Gradient Boost, KNN, and Gaussian NB demonstrated acceptable performances higher than 70% in specific cases. The confusion matrices for MLP and Extra Trees are shown in Fig. 5.12. It can be observed that these classifiers with hand-crafted features can discriminate some classes better than other. The LPS class was predicted well for both models. This class has very discriminative covariance and geometric features. Regarding the FH and TR classes, MLP predicts better than Extra Trees. MLP showed a better performance than Extra trees in terms of precision, recall, and accuracy. However, for classes related to traffic signs (SNP, IRS, SGW, and combined signs) both models showed low performance on the classification. This is mainly due to the similarities between classes, missing points and noise during scanning. Also, the covariance features extracted might not

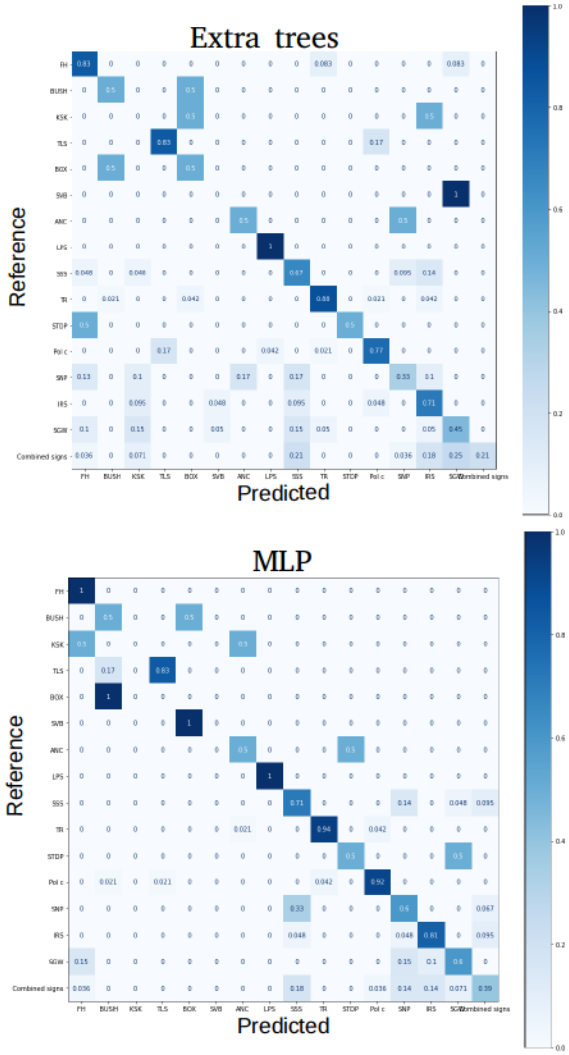


Figure 5.12: Confusion matrices for Extra Trees (Top) and MLP (bottom) classifiers on Tall-16 dataset. The x-axis denotes the predicted labels while the y-axis denotes the groundtruth labels.

be sufficient to distinguish these classes using the previous classifiers.

5.4.4 Classification on Flat-9

The classifiers demonstrated good performance for both tall objects and flat objects, as shown in Table 5.5 using the Flat-9 dataset. Notice that the ensemble soft voting classifier again achieved the maximum accuracy value of 93.52% as compared to other classifiers using the scaled PCA for feature scaling, which was 3% less than in the Tall-16 dataset. Like its previous behavior in Tall-16, the Hyperband algorithm again showed the second-best performance value of 81.33% using the scaled-PCA. The Extra Trees algorithm demonstrated better performance than the rest of the classifiers in the classification of Flat-9 when applied scaled, min-max, or min-max PCA as feature scaling.

5.4.5 Classification on Tall-25

This dataset is derived from applying data augmentation to 25 different Tall classes from LiSurveying. We balanced this dataset proportionally by increasing each class with a value of 375 samples. The classifiers showed a decreasing performance when applied to Tall-25. This can be due to similarities between object classes and occlusion of important features. The MLP classifier has achieved a maximum accuracy of 73.53% on this dataset, followed by the ensemble soft voting classifier and Hyperband L2 classifier with 68.09% and 68.47%, respectively. The rest of the classifiers showed poor performance at less than 68%.

5.4.6 Classification on Tall-14 and Flat-10

In Tall-14, after data augmentation we aim to overcome the problem of similar object classes presented in Tall-25. In this experiment, we merged all traffic sign classes and pole classes into two unique classes: Signs and Poles. This helps to reduce the computational complexity and increases the performance of the models. The confusion matrices in Fig. 5.13 demonstrate that by balancing the datasets with data augmentation operations on the existing dataset, models can significantly improve the classification task. Again, the MLP classifier showed the highest performance, achieving an accuracy of 91.38% on this dataset, followed by the ensemble soft voting classifier, Hyperband classifier, and Random Forest with 89.17%, 85.48%, and 84.03%, respectively. The confusion matrices for the voting classifier and MLP classifier in the Flat-10 dataset are shown in Fig. 5.13. It can be seen from the diagonal that data augmentation benefits the model's classification performance. Since our hand-crafted

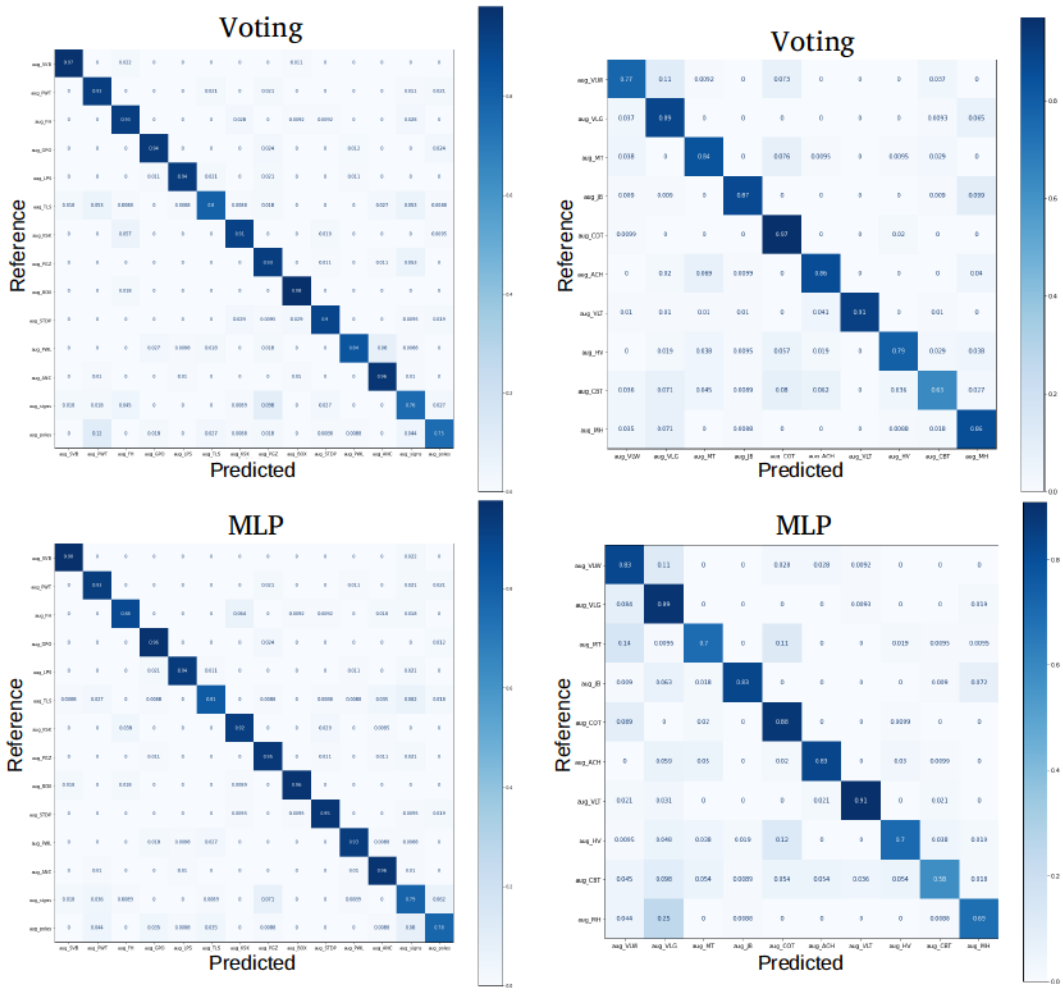


Figure 5.13: Rows 1-2: Confusion matrices for Voting classifier and MLP classifiers on Tall-14 dataset. Rows 3-4: Confusion matrices for same classifiers on Flat-10 dataset.

feature extraction modules are defined specifically for tall objects or flat objects but not both object types, conventional machine learning methods were not tested on Merged-39 and Merged-39* datasets.

5.4.7 Classification using PointNet and PointNet++

Experimental results on our datasets using PointNet and PointNet++ are summarized at the bottom of Table 5.5. In our experiments, the PointNet model achieved a classification accuracy of 41.37% on the Tall-16 dataset, which shows low performance compared to other classifiers. This is because PointNet took the point input directly instead of using hand-crafted features. When using the Flat-9 dataset, PointNet obtained a performance of 17.21%, demonstrating very low classification performance on

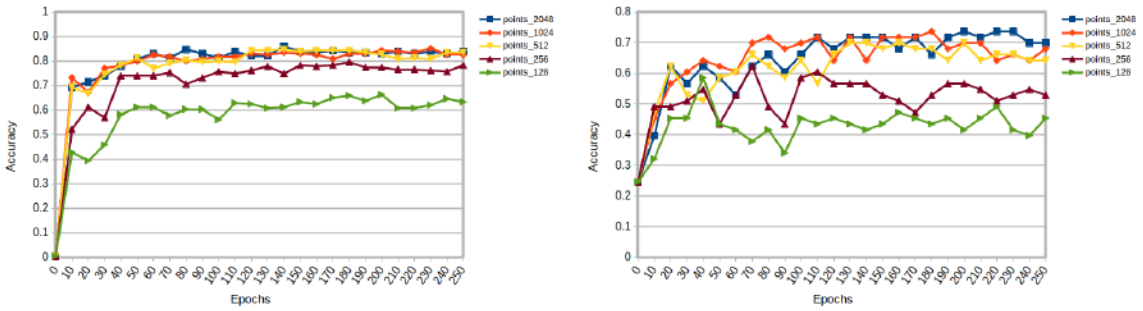


Figure 5.14: Tall-16 and Flat-9 classification using PointNet++ with different number of inputs points during training.

object classes with flat shapes. After applying data augmentation, PointNet showed an improvement on Flat-10 and Merged-39. However, the performance is still inadequate for object classification. Several factors affect the performance of PointNet. First, its architecture, designed to operate on each point independently, limits its capability to capture local features at different scales. Second, the artifacts, such as noise, present in the samples cannot be discriminated effectively and are learned by PointNet as valid features, causing confusing at the classification step.

We further evaluated PointNet++ in some subsets of LiSurveying. It can be observed from Table 5.5 that our setup of PointNet++ outperforms PointNet by a large margin on both the Tall-16 dataset and the Flat-9 dataset, achieving 83.76% and 73.58%, respectively. We evaluated this model using different SA layer parameters during training to better investigate the performance. We also conducted a comprehensive ablation experiment with PointNet++ to examine its behavioral performance with different input point sizes during the training of Tall-16 and Flat-9 datasets.

Effect of reducing points during training

We investigated the effect of downsampling the input points on the classification performance. We tested different N_{points} values (128, 256, 512, 1024, and 2048) and trained each model for a maximum of 250 epochs as shown in Fig. 5.14. As expected, reducing the number of points during training can gradually affect the classification of PointNet++. It can be observed that when evaluating PointNet++ on Tall-16, the model revealed an increase in performance and ability in generalization by receiving a greater number of points in the input layer. We observed that the models trained with N_{points} = 2048, 1024, and 512 showed a similarly good performance, which demonstrates the robustness of PointNet++ using these values. This method can preserve important points with previous parameters to distinguish the classes in

Tall-16. When the N_{points} parameter was reduced to 256, the performance of the model dropped by around 5%, and continued to drop when the value was reduced to 128 points. The accuracy of PointNet++ also dropped and it showed unstable behavior in Flat-9, which contains classes with flat object point-clouds. Important information is lost when flat objects are downsampled randomly. This is because the distribution of points is represented along the horizontal axis with respect to the ground. Using 2048 points, PointNet++ can obtain an accuracy close to 73%-74% in Flat-9 and has slightly similar classification performance with 1024 and 512 points. When downsampling the points less than 512, the classification network drops in accuracy by around 20%, which shows the sensitivity of this network when classifying flat objects with a low number of points.

Effect of SA layer dimensions and Dropout

PointNet++ architecture has three SA (set abstraction) layers, and we investigated the effect of MLPs dimensionality within the first and middle SA layers on the classification. Also, we investigated the effect of the dropout layer (the fraction of the input units to drop) between each pair of fully connected layers as it is important when performing classification with this architecture. Both effects are reported in Table 5.7, We trained PointNet++ with different configurations on Tall-16 dataset. In all the experiments we used a radius of 0.2 in the first SA layer, and 0.4 in the second SA layer. We can observe in Table 5.7 that PointNet++ achieved 83.75% accuracy in the first experiment. Second experiment has the same configuration on SA layers and MLPs as the previous experiment, but with a different dropout parameter of 0.3 on each layer. The classification accuracy increased to 84.19% when we reduced the number of input units between layers. In the third experiment, we observed that with a dropout factor of 0.5, when the dimensionality of the MLPs in the first SA layer was changed from ($64x64x128$) to ($128x128x128$), PointNet++ showed a better performance of 85.47%. However, the last experiment showed that modifying the second SA layer results in better classification performance (86.75%) than the previous ones.

Table 5.7: Effect of SA layer dimensions for PointNet++ on Tall-16 dataset.

Exp	SA1	SA2	SA3	dp	Acc
1	$64x64x128$	$128x128x256$	$256x512x1024$	(0.5, 0.5)	0.8376
2	$64x64x128$	$128x128x256$	$256x512x1024$	(0.3, 0.3)	0.8419
3	$128x128x128$	$128x128x256$	$256x512x1024$	(0.5, 0.5)	0.8547
4	$64x64x128$	$128x256x256$	$256x512x1024$	(0.5, 0.5)	0.8675

5.5 Conclusion

We introduced a 3D urban asset dataset for point-cloud classification and detection. The dataset includes different dense scenes with millions of points and uncommon urban object classes. Our LiSurveying dataset is designed for land surveying and site analysis applications, and can be used to supplement other 3D point-cloud datasets to provide richer scene contents. LiSurveying can support the evaluation of different algorithms because of the diverse characteristics, such as number of points, number of objects, number of classes, as well as object shapes and sizes in complex outdoor environments.

Preliminary baseline experiments were conducted for point-cloud classification on seven subsets of the LiSurveying dataset using hand-crafted features with several classifiers, PointNet and PointNet++. Existing datasets lack the uncommon object classes that we contributed. Different approaches, e.g., computer vision, machine learning, and deep learning, can be adopted by researchers based on their application requirements, volume of point-cloud available, and the object features to be detected. No matter which approach is selected, our dataset can be useful as it contains uncommon urban assets not covered, or inadequately covered, in existing datasets.

In future work, we will annotate more object classes in LiSurveying, and compare and analyze different 3D object detection, semantic segmentation, and urban asset classification algorithms.

Chapter 6

LiDAR object detection and key-points for WAM

6.1 Introduction

Conventional surveying processes require accurate collection of geomatics for development and planning of urban cities or towns. Highly skilled land surveyors seek out different urban physical objects on the surface or under the ground in their typical routines. Hydrants are one of the most significant and common objects due to their usefulness and importance in daily life. They are typically installed with underlying water networks in communities, streets, or avenues. This urban object serves an important function in providing secure and fast access to a water supply system, and can assure steady water supply to firefighters in the event of a fire. Governments and communities are spending increasing resources to train and educate the public about the importance and value of this essential equipment. In a large fire outbreak scenario, the difference between human loss and loss of essentials is determined by the time firefighters localize and access water from an available hydrant in-situ. Land surveying tasks include measuring the relative position of the fire hydrants along streets and avenues. In British Columbia, NFPA-24 and NFPA-291 provide standards for fire hydrants in terms of color (blue, green, orange, and red), class (AA, A, B, and C), and flow (1500, 1000-1499, 500-999, and less than 500 gallons per minute (GPM)), respectively. The traditional surveying techniques involve manual tasks such as calculations and production of plans, to determine an accurate position of the objects. For example, in a standard surveying procedure, a surveyor assists the geographical area under study with accurate devices such as a Global Navigation Satellite System (GNSS) receiver, prismatic compasses, clinometer, among others. The surveyor localizes the hydrant and collects latitudinal/longitudinal coordinates of interest points (e.g., the

stem nut, side outlet, or chain), height and size, and additional information such as area, volume, type, color, etc. The collected information is used to generate CAD drawings ensuring that every hydrant in a district performs properly and reliably when an event occurs. Although these conventional surveys and proper assessments can provide accurate geomatic data, they may involve manual tasks requiring a high level of expertise to execute. Furthermore, the presentation of the geomatic information should be rendered either graphically in the form of plans or numerically in the form of tables, and this relies on expensive software tools. These factors, and the rapid expansion of cities and towns, result in a time-consuming, restricted, and expensive process. Therefore, automation strategies and surveying tools are needed to increase productivity in detecting urban objects such as hydrants. The use of LiDAR in urban planning is currently an emerging alternative to automate conventional processes and retrieve accurate information from the earth's surface using Machine Learning and Computer Vision algorithms on point-cloud data.

We propose two methods: first, a learning-based object localization method of fire hydrant urban objects in dense TLS point-cloud outdoor scenes, and second, an outlet key-point detection for centroid point estimation in order to automate CAD drawing generation. The proposed methodology considers challenging scenarios where multiple fire hydrant objects are present across extensive TLS point-cloud data acquired with variate density of points.

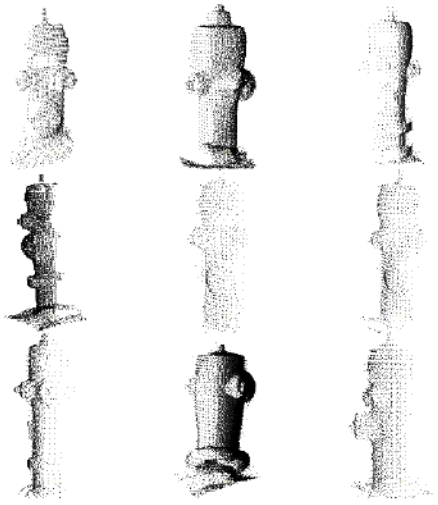


Figure 6.1: Multiple hydrant objects from our FH-LiDAR dataset.

6.2 Related Work

6.2.1 Object detection approaches on 3D

In the last decades, several 3D descriptors have been introduced to recognize objects in 3D space. These are known as 3D descriptor-based recognition methods. These algorithms are generally divided into two main categories, local descriptors, and global descriptors. Local descriptors are mainly focused on characterizing the neighborhood of a key point by extracting features such as curvatures, histograms, and gradients. While local descriptors obtain relevant features on a specific area, global descriptors extract features from the entire object space in a single feature vector. These methods are directly associated with the quality and density of point-clouds. Local descriptors included algorithms such as Spin Images, 3D Shape Context, Eigenvalues based descriptors, Distribution Histograms, Histogram of Normal Orientation, Intrinsic Shape Signatures, Point Feature Histogram, Fast Point Feature Histogram, Radius-based Surface Descriptor, Normal Aligned Radial Feature, Signature of Histogram of Orientation, Unique Shape Context, Depth Kernel Descriptor, and Spectral Histogram. Widely used global descriptors are Point Pair Feature, Viewpoint Feature Histogram, Global Structure Histograms, Shape Distribution on Voxel Surfaces, and Scale Invariant Point Feature. However, descriptors methods have some weaknesses, such as sensitivity to occlusion, deformation, outliers, and point density variation, which make them inconvenient for real-world applications.

Recently, several approaches have used Deep Learning methods to detect objects directly from point-clouds or a combination of multiple sensors, including optical images and point-clouds. These object detector networks are divided into two main categories: region proposal-based and single-shot algorithms. The main idea of these approaches is to generate a 3D bounding box around each detected object from an input point-cloud. Region proposal-based methods have demonstrate high performance on the detection task, and are divided into three categories: multi-view based [181–186], segmentation-based [187–192] and frustum-based methods. Single-shot methods are also divided into three categories: bird's eye view-based (BEV), discretization-based and point-based methods. All of them are differentiated by performance time, modality and accuracy.

MV3D [181] is a pioneer among the works in the multi-view based category by combining multiple data sources (e.g., LiDAR, optical images, and BEV) and projecting them into a feature map to extract accurate 3D bounding boxes. Similarly, Ku *et al.* [182] combined equally features vectors from both BEV and optical image views computing cropping and scaling operations to be fused by using elementwise

mean-pooling. Subsequently, Liang *et al.* [183] presented an approach that combines images and LiDAR points with different resolutions to find corresponding features. They obtained dense BEV feature maps by associating 2D features from the image into the BEV plane. Later, Liang *et al.* [184] introduced a multi-task framework to exploit different tasks such as detection, ground estimation, and depth completion to improve the learning of the network. These approaches demonstrate high accuracy object detection experimental results, but a major limitation is the high computational cost of manipulating different multi-source data and fusion modalities.

Segmentation-based methods are supported by operations related semantic segmentation to remove not interest points and reduce the computational cost. The author in [187] introduced a method that first segments the objects from 2D images and projects each pixel of a mask into point-clouds to subtract background points. Then, they applied a criterion to reduce redundancies, and finally, the proposals were extracted from the remaining points for bounding box inference. A similar approach is PointRCNN [188], which fused semantic features and local spatial features for bounding box inference. Subsequently, Li *et al.* [192] presented an improved version of PointRCNN by proposing a three-branch backbone network to handle non-uniform density points. Jesus *et al.* [189] introduced a two-module object detector. The first module includes an R-GCN that gathers meaningful features between a point on each proposal. The second module, namely per-frame C-GCN, obtains contextual information between proposals on each PCD frame. These segmentation-based approaches are suitable for complex point-cloud scenes that include challenging factors such as high occlusion and crowded objects and are faster than multi-view based methods.

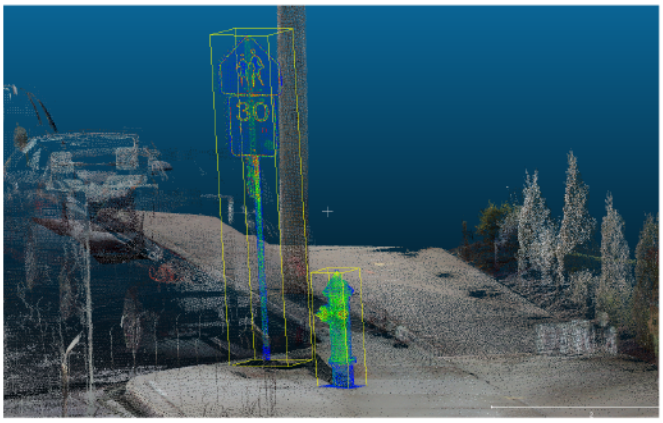


Figure 6.2: Manual annotation of urban objects with a 3D bounding box.

Likewise, frustum-based methods are frameworks that fuse candidates bounding boxes from 2D object detector networks and extract 3D frustum proposals (set of

points) from those candidates. Different methods are found in this category. For example, F-PointNets generates a frustum region for every candidate on the image and then learns point features on the frustum using 3D networks such as PointNet. Point-SENet integrates the PointSIFT module and scaling factors invariant to object scaling. It demonstrates significant results in outdoor environments. PointNet++ [92] overcomes the problem of missing information in between points by using a sampling layer, grouping layer, and the PointNet-based learning layer. This setup can capture the geometric features of the neighborhood of each point. However, previous approaches involved multiple step-by-step processes due to the incorporation of 2D detectors and frustum generation, which limited their performance.

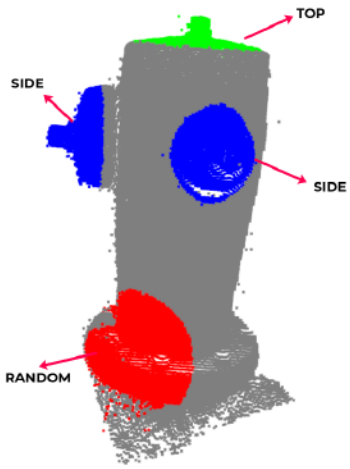


Figure 6.3: Hydrant object from our dataset annotated with Top Outlet class (green), Side Outlet class (blue), and Non Outlet class (red).

6.3 Methods

We implemented an end-to-end pipeline to address two problems, object detection and key-point feature detection, directly from outdoor point-cloud scenes on hydrant objects. Different than current object detection approaches, our pipeline not only outputs a 3D bounding box with a class of the target object but also an interest key point on the predicted object. When surveying a hydrant, a surveyor first localizes this asset and then measures the centroid point closest to the ground surface with accurate expensive tools. We demonstrate that our proposed strategy can handle complex 3D point-cloud scenarios where hydrant object instances contain point-cloud variation, noise, and partial to heavy occlusion. Therefore, we propose a pipeline with two stages that relies solely on point-clouds and several robust techniques, contributing to a more

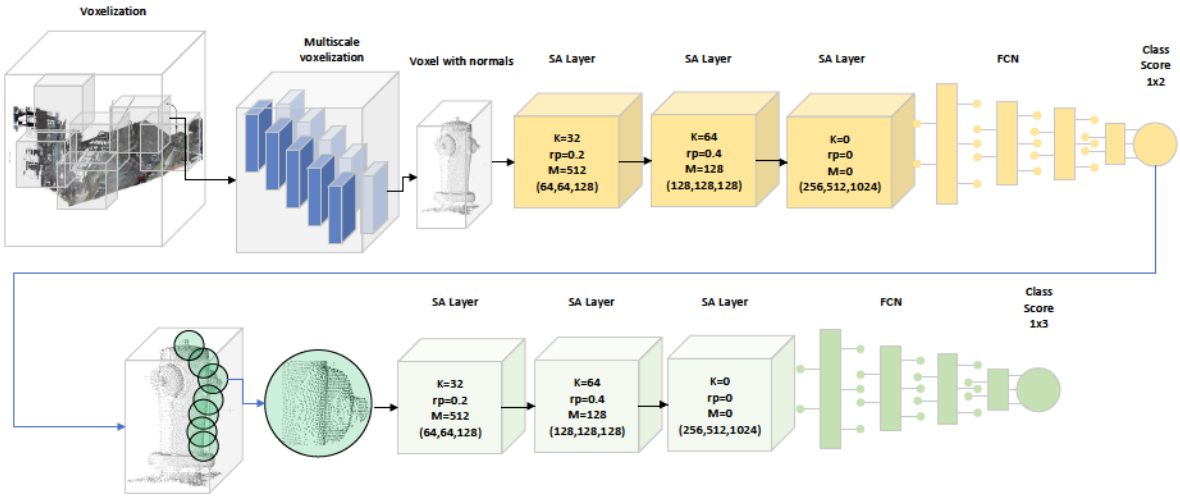


Figure 6.4: Our proposed pipeline with different stages.

effective refinement of the final predictions. In the first stage, for object detection, we introduce a dynamic voxelization approach to partition the raw point-cloud into fixed voxels based on prior information from training data. Then, we adopt PointNet++ to train a model using point-clouds instances in order to distinguish between a fire hydrant class and an unknown class. The predicted class and 3D bounding box with the hydrant object are inputs for the second stage of our pipeline. In the second stage, our goal is to localize key point features of the hydrant, such as the top outlet or side outlet. Thus, we implement an algorithm to generate training data using an iteratively searching strategy with KD-trees and previously annotated ground truth key points on hydrant point-clouds. Then, we train a classifier to localize the key points, such as top outlet or side outlet, that support the final centroid point estimation.

6.3.1 Object database preparation

Our dataset was acquired with an ultra high-speed time-of-flight device enhanced by a Waveform Digitising (WFD) technology Leica ScanStation P20 sensor. Point-clouds were acquired in multiple locations with 360 degrees horizontal and 270 degrees vertical in British Columbia, specifically in the areas of New Westminster (NW), Marine Way (MW), and Nanaimo (NN). The sensor's scan wavelength is 808 nm (invisible) / 658 (visible) with a range of 120 m; 18% reflectivity (minimum range 0.4 m) and scan time and resolution (12.5mm @ 10m). A total of 41 hydrant objects were localized using a CAD file related to the point-cloud scene. Then, each hydrant was annotated manually with a 3D bounding box and a class

using CloudCompare software (Fig. 6.1). A bounding box contains the following parameters: center coordinates (cx, cy, cz), height, width, and length of box (h, w, l), box corners coordinates (m-by-n matrix, with m=8 and n=3), maximum bounds for geometry coordinates $(maxx, maxy, maxz)$, minimum bounds for geometry coordinates $(minx, miny, minz)$, and unique ID. It also contains object coordinates points (obj_x, obj_y, obj_z), rgb color and the intensity value of each point as shown in Fig. 6.2. For the key-point detection approach, we annotated ground truth coordinate points (u_x, u_y, u_z) on the hydrant's outlets surface with a class o_{class} Top Outlet ("TO"), Side Outlet ("SO"), or Non-Outlet ("NO"), as shown in Fig. 6.3.

6.3.2 Scene voxelization preprocessing

A dense outdoor point-cloud scene represents a large volume of data with millions of points that requires pre-processing to reduce the computational cost of processes. Thus given an input large scale point-cloud with N points denoted by $\boldsymbol{S} = \{\boldsymbol{s}_i \in \mathbb{R}^{3_{x,y,z}} |_{i=1,\dots,N}\}$ we attempted to partition $\boldsymbol{S}$ into non-overlap voxels by a quantization parameter $d = [d_w, d_l, d_h]$, where w, l, and h define the width, length, and high parameters of each voxel. The partition process is denoted as $\left\{ \bar{v}_i = \left(\left\lfloor \frac{x_i - Xmin}{d_w} \right\rfloor, \left\lfloor \frac{y_i - Ymin}{d_l} \right\rfloor, \left\lfloor \frac{z_i - Zmin}{d_h} \right\rfloor \right)_{i=1,\dots,N} \right\}$, where $\bar{v}_i$ is the voxel index integer used as key to access related subsets points (subscenes) within each voxel. Based on our experiments with different $[d_w, d_l, d_h]$ parameters, [10,10,50] generates the best results and guarantees that each hydrant is within one scene without loosing information. Then, within each $\bar{v}_i$ we applied a multiscale voxelization process for object detection.

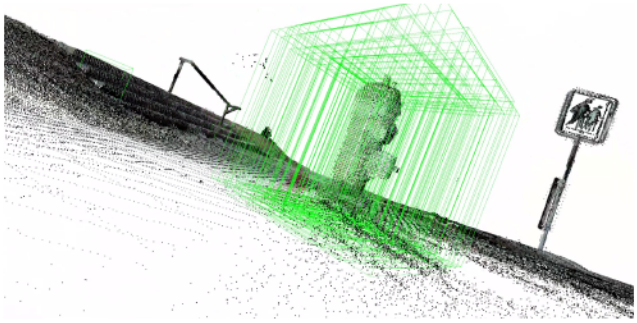


Figure 6.5: Generate voxels over raw point-cloud scene with our multiscale voxelization approach. We showed the generated voxels with an intersection over union (IoU) threshold of 90% respect to the ground truth.

6.3.3 Subscene multiscale voxelization

After scene voxelization, we consider a subscene multiscale voxelization in order to obtain accurate bounding box predictions as, shown in Fig 6.5. Each $\boldsymbol{H} = \{\boldsymbol{h}_i \in \mathbb{R}^{3_{x,y,z}} |_{i=1,...,N}\} \in \bar{v_i}$ is voxelized into overlapping voxels by a quantization parameter $d' = [d'_{w'}, d'_{l'}, d'_{h'}]$, Δ_s, and q_s, where w', l', and h' correspond to the maximum extend axis-aligned bounding box width value from each class in our training data. Δ_s is the interval parameter applied on each axis and q_s denotes the step size. The smaller the q_s is, the finer the voxelization but higher computational cost. On the contrary, a higher q_s reduce the processing time but might lead to high redundancy voxel generation. An example of redundancy can be two neighbor voxels containing half and half structure of the same object in the scene. The whole multiscale process is denoted by $\left\{ \begin{array}{r} \\ \bar{v'_i} = \left(\left\lfloor \frac{x_i - x_{min} - \Delta_s}{d'} \right\rfloor\right) \\ \\ \\ \end{array} \middle| \begin{array}{r} i = 1, ..., N \\ d'_{w'} = 0, ...w', \frac{w'}{q_s} \\ d'_{l'} = 0, ...l', \frac{l'}{q_s} \\ d'_{h'} = 0, ...h', \frac{h'}{q_s} \end{array} \right\}$, where $\bar{v'_i}$ is the voxel index integer used as key to access related subsets points that need to be classify within each subvoxel.

6.3.4 Hydrant localization from raw scenes and centroid point estimation

After the subscene multiscale voxelization stage, we adopted PointNet++ to classify the point sets on each $\bar{v'_i}$ in hierarchical fashion as shown in Fig. 6.4. This method has shown acceptable results in applications that require point-clouds 3D object classification or segmentation. It leverages small neighborhood sets of points at multiple scales to extract local and high-level features. This process permits both robustness and detail capture on point-cloud data. The hierarchical structure consists of a set of abstraction (SA) levels. Each abstraction level has three key stages (sampling, grouping, and MLP layer) that extract multiple scales of local patterns and combine them logically according to local point densities. At each level, a set of points is processed and abstracted to output a set with fewer elements. Given the points $\boldsymbol{h}_i \in \mathbb{R}^{3_{x,y,z}}$ within $\bar{v'_i}$, these are subsampled using an iterative farthest point sampling method to obtain query points. For the grouping stage, a ball query method is used to localize an upper limit k neighbors of each query within a radius r; each group corresponds to a local region. Each local region is abstracted by its centroid and local feature

that encodes the centroid's neighborhood. The process is repeated until obtaining the features on $\bar{v}_i'$. Finally, the obtained features are transformed by a multi-layer perceptron (MLP) network, followed by three fully connected layers and a softmax function to obtain the probability class score.

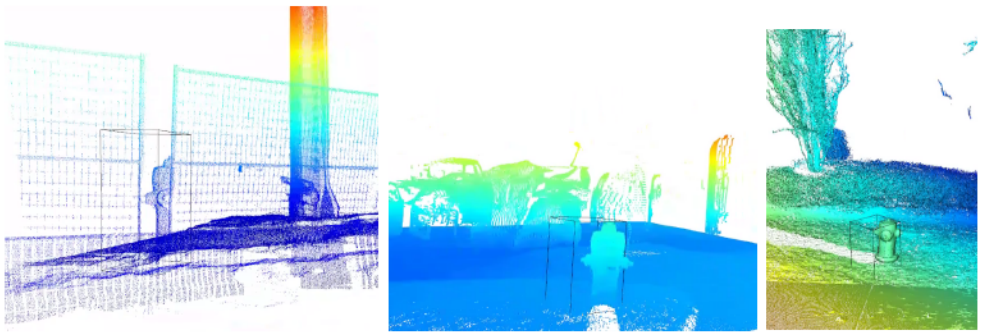


Figure 6.6: Predicted bounding boxes with FH class.

6.3.5 Centroid point estimation

After predicting the FH class within a voxel as shown in Fig. 6.6, we use the K-Dimensional Tree (KD-Tree) algorithm with k-nearest neighbors (KNN) search to iterate over the set of points and extract neighbor subsets within a radius with respect to a query point. Then, the subsets are input to the second PointNet++ model to predict whether the points represent an outlet. After classifying voxels as "TO," "SO," or "NO," we wanted to select the best "TO" and "SO" predictions to measure the hydrant's centroid accurately. Then we can measure the x, and y centroid by averaging the points in the top outlet surface. In the case that the top outlet is not presented or detected in the point-cloud, the side outlet predictions can also be used to estimate the hydrant's centroid point. Note that the top outlet part of a hydrant is normally centered to itself. After conducting our experiments, we were able to obtain accurate centroid results in many of the cases. Fig. 6.7 (a) and (b) show the predicted top outlet by our model, and (c) and (d) show the final estimated centroid point (red color) after averaging the x and y coordinates. Note that we also tried averaging the whole hydrant object or horizontal slices. However, none of these other approaches were successful because different hydrant point-clouds might have missing points, noise, or artifacts in between the points that cause averaging to fail.

6.3.6 Implementation details

The classification network based PointNet++ receives in the SA stage a point-cloud with spatial coordinates and face normals associated to $\bar{v}_i'$. The SA stage consists

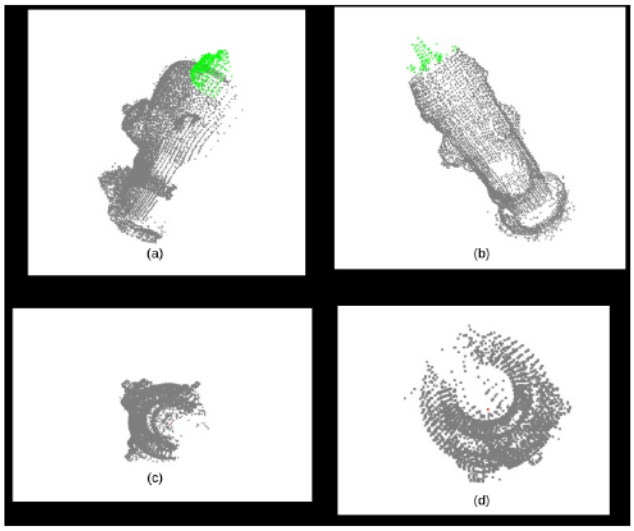


Figure 6.7: Centroid point estimation. Hydrant's points classified as top outlet points (Top). Results of estimated final centroid point (Bottom).

of three SA layers. In the first SA layer, 512 sampling points are selected from the input and local regions are delimited by determining 32 nearest neighbors of each point within a 0.2 radius. From these local regions in the current SA layer, features are extracted using MLPs (64,64,128). Then, the extracted features are summarized for 512 regions with max-pooling. Similarly, in the subsequent SA layer, features are extracted from 128 local regions with MLPs (128,128,128). The third SA layer, received the remaining points for the extraction of features in a unique local region. The global features in this SA layer are extracted with the MLPs (256,512,1024) with max-pooling summarization. The final 1024 dimensional global feature is linked to the two label classes (fire hydrant (FH) and non fire hydrant (NFH)) through MLPs layer containing (1024, 512, 256) and two neurons. In all layers except last, ReLU (512, 256,), batch normalization (512, 256,) and dropout (0.5, 0.5,) are applied. Then, in the case that the previous model predicts an FH class object, the set of points within the bounding box are passed through an additional trained model to classify small patches on Top Outlet (TO), Side Outlet (SO), or Non-Outlet (NO). The additional model has the same parameters as for FH and NFH classification. However, the final 1024 dimensional global feature is linked to three classes through MLPs layer containing (1024, 512, 256) and three neurons.

Table 6.1: Comparison of 3D object classification results using PointNet++ on the FH LiDAR dataset (2-classes) with different configurations

Model	N Points	Augmentation	Accuracy
PointNet++	256	No	46.15
PointNet++	512	No	54.36
PointNet++	1024	No	61.54
PointNet++	2048	No	76.92
PointNet++	4096	No	76.92
PointNet++	256	Yes	53.64
PointNet++	512	Yes	72.16
PointNet++	1024	Yes	99.48
PointNet++	2048	Yes	99.61
PointNet++	4096	Yes	99.84

6.4 Experiments

In table 6.1 we demonstrate the results on the FH-LiDAR Dataset along with different configurations in terms of training data, data augmentation, and the number of input points. In the experiments, we evaluated PointNet++ on the FH-LiDAR dataset that contains 41 hydrant point-clouds dispersed in three different urban outdoor scenes from British Columbia. This diversity of configurations in the dataset allows us to validate the performance of PointNet++. Using the original dataset, we observed that the maximum accuracy obtained for classification is 76.92% with 4096 input points. The model provides the same performance even by sampling the input points to 2048. However, the performance decreases when sampling the input points from 1024 to 256 (61.54% and 46.15%, respectively). Compared to the original dataset, the accuracy increases to a maximum value of 99.84% when training the model with the augmented FH-LiDAR dataset. For key-point detection, we trained a model with batch size 32, a learning rate of 0.001, and a maximum of 1024 points in each object without augmentation. We got 99.10 % training accuracy and 98.42% testing accuracy after training for 50 epochs.

6.5 Conclusion

We propose a pipeline to achieve object detection and key-point estimation of hydrant objects on urban point-clouds scenes. Our object localization method consists of a multiscale dynamic voxelization strategy to extract hierarchical features within generated voxels. Our pipeline, integrating multiple modules of learning-based models, demonstrates robust detection and recognition accuracy on point-clouds. In future work, we will explore optimization to further improve the time and accuracy performance.

Chapter 7

UAV feature extraction for WAM

7.1 Introduction

Route planning and mapping play important roles in maintaining the safety and control of major highways and avenues in rural areas. As-built surveys are typically conducted during the construction process to ensure that procedures are carried out according to the design and that roads are built exactly as planned. The subsequent stage involves surveying completed roadways, where surveyors use a combination of high-performance total stations and real-time kinematic (RTK) GPS systems to obtain location and boundary information. In the past, surveying was typically a manual process that involved using a theodolite and steel band to measure angles and distances between points. A theodolite is a precision optical instrument used to measure angles between visible points on horizontal and vertical planes. The total station simplified many traversing procedures for surveyors by integrating an electronic distance measurement device (EDM) into the theodolite. Later, the introduction of satellite positioning systems (GPS, GLONASS, Galileo, BeiDou, NavIC, QZSS, and SBAS) provided surveyors with a more accurate strategy for conducting survey activities, which benefitted many industries. Currently, the combination of RTK+GPS is widely used in the surveying industry due to great performance in providing centimeter accuracy in the field. Although these conventional surveys and assessments can provide accurate geomatic data, they are predominantly manual tasks that require domain-expertise and expensive specialized equipment. The completion time and operation cost of a land survey is determined by the size of the area and different features of interest; this is a repetitive task for surveyors, who have to pick points in large areas covering kilometers. Moreover, environmental factors, e.g., tropospheric activities, canopy covers, and buildings, can cause signal interference, affecting the accuracy of the survey.

In view of the rapid expansion of municipalities, this manual process becomes time-consuming, restricted, and limited to skilled labor. Purpose-built tools have emerged that allow survey tasks to be carried out using imagery or point cloud data. Such popular commercial software suites include Quick Terrain Modeler, AutoCAD Civil 3D (Autodesk), Mars 7, Ecopiatech, and TopoDOT. These suites allow users to accurately extract a variety of lines and points features data, e.g., side walks, traffic signs, or even lines on the road, as in our case study. However, these commercial tools also show limitations that must be considered. First, they are licensed and dependent on commands by the user, either through clicks or with touch screens, which limits them to performing fully autonomous tasks. The user typically has to deal with loading and then manipulating the data using rotate, zoom, and translate operations until the target zone is found. Using sidewalk extraction as an example, the user typically clicks near the sidewalk and then features in the area are extracted to find the sections connected to the same sidewalk. Although the results are usually accurate, they always require user intervention to continue extracting features in the rest of the scene. Second, performance is impeded by increasing input data, which slows down processing or even aborts commands due to excessive memory consumption. And third, the resulting feature lines or points must be refined by the user at various locations in the scene. More recently, drone mapping systems produce high-quality data, allowing detailed map representation of the scene from a different perspective, which can cover much larger areas of interest (AOIs) than conventional surveying. Object detection and object classification are standard terms in Computer Vision (CV) and have gained attention in remote sensing applications. These concepts aim to determine whether data contains a target entity, match, feature, signal, pattern, or activity. Related research has gained attention in the last decades and its studies have generated large amounts of digital multimedia data, e.g., time series, images, videos, or point clouds, in numerous industrial sectors and scientific fields. Object recognition and object detection typically provide instances belonging to a given category or class, returning the spatial and/or extent location of a target object, e.g., object class, 2D bounding box, or both object class and location simultaneously. Semantic segmentation as a further extension of object classification, labels every pixel in the image with a corresponding object class label. Different techniques are adopted depending on the applications. The integration of artificial intelligence (AI) in land surveying using drone imagery has the potential to revolutionize the field. By automating previously manual tasks, AI algorithms can quickly and accurately extract important data points such as topography, elevation, and boundary lines. The speed and efficiency of AI algorithms make them ideal for processing large amounts

of data and identifying patterns and anomalies that may be difficult for humans to detect. Moreover, drones are an efficient tool for land surveying, capable of covering large areas in a short amount of time. This makes drones ideal for surveying large projects such as construction sites or infrastructure developments. AI algorithms can analyze the data collected by drones to detect changes in terrain, indicating potential hazards such as landslides and erosion. Human expertise and knowledge are still crucial for interpreting and analyzing the data generated by AI algorithms. Overall, the integration of AI in land surveying using drone imagery has the potential to greatly improve efficiency and accuracy, but it should be viewed as a tool to augment the skills of human surveyors, rather than replace them. For our goal of extracting linear features in an image, which may have slopes and be continuous in extent, detecting lines with bounding boxes is not feasible. On the other hand, semantic segmentation allows the classification of each pixel belonging to the line in the entire extension of the image regardless of the trajectories or directions. In this sense, aerial imagery with computer vision algorithms can offer an automatic workflow for a variety of land surveying tasks. Our framework involves multiple stages that are responsible for extracting and mapping target lines in high resolution drone images, which results in a Computer-aided design (CAD) that surveyors require. A novelty is the framework itself, where each stage is crucial to achieving accurate extraction of lines, and the benefits are seen in the processing time and precision of lines in large coverage areas without user intervention. In our method, we take advantage of the quality of high-resolution images to extract pavement lines with greater precision, less time, and less computational cost than current methods. We use a semantic segmentation algorithm for the initial estimation of the lines, and consequently a stage of numerous cascading optimized filters to clean, refine, and merge the target lines. We performed experiments with different semantic segmentation algorithms (and configurations) to evaluate the best one for this task.

7.2 Dataset creation for road marking surveying

We demonstrate the feasibility of DL semantic segmentation and image processing for accurate road marking surveying using high resolution aerial images. The existing datasets related to our problem domain lacked the additional information needed to validate the tasks of road marking surveying. Our dataset is distinguished from existing urban datasets in many important aspects. The ground truth data extraction is based not only on pixel-wise annotations, but also on a validation of the annotation with line information in the physical world. This validation can be conducted by

comparing the resulting lines with the CAD engineering drawings provided by domain experts in the area, which represent the real, precise, and reliable geospatial locations of the point features and linear features in the ground coordinates system. Ground coordinates are measurements within 1 cm accuracy that are taken using GNSS RTK receivers on the actual surface of the earth. In this way, the data obtained in each stage of our framework can be analyzed and compared more rigorously with real-world assets, either image space or line space.

Glover Road dataset

The survey area selected was in Langley, specifically Rawlison Crescent and 240 St., British Columbia, Canada. The area used for the study covers 5.5 km^2 of land with a diverse landscape composed of grass wetlands, forage grass, palmetto wet and dry prairies, pine flatwoods, surrounding mountain and valley views, and interconnected farms throughout, as shown in Fig.7.1. The area contains 2.7 km of two-way highway that has painted traffic signals that divide and delimit the edges of the road, as illustrated in Fig.7.2. Two sets of solid double yellow lines that are two or more feet apart sometimes appear as a road marking. Such lines indicate that passing is not permitted. Edge-lines help drivers stay on the road during low visibility conditions, such as fog, rain, and snow. The road marking along Rawlinson Crescent and 240 St. presents diverse conditions due to external factors, which include damages generated by third parties, atmospheric effects (rain, pollution, dirt particles, and gases, among others), vandalism, and/or road accidents. These factors that lead to the deterioration of the signal are reflected in the reduction of retro-reflectivity and in the partial or total destruction of these elements affecting the display of information on the roads. The previously listed factors directly affect the patterns in 2D images, representing challenges for semantic segmentation models. The dataset is comprised of two high resolution images, LRS1 and LRS2, taken during a mapping mission in the study area. The image LRS1 has 23022×55146 pixels in Tag Image File Format (TIFF), containing 3 Bands (RGB) and 8 bits pixel depth. LRS2 shares similar image and spatial information with LRS1 but has a size of 66520×26296. High resolution imagery data (LRS1 and LSR2) was acquired in the spatial reference NAD (North American Datum) 1983 CSRS (Canadian Spatial Reference System) UTM (Universal Transverse Mercator) Zone 10N with a PHANTOM 4 RTK (Real-time kinematic positioning) as specified in Table 7.1).

Figure 7.1: Example of different environments encountered in the Glover Road Area, and respective CAD files.

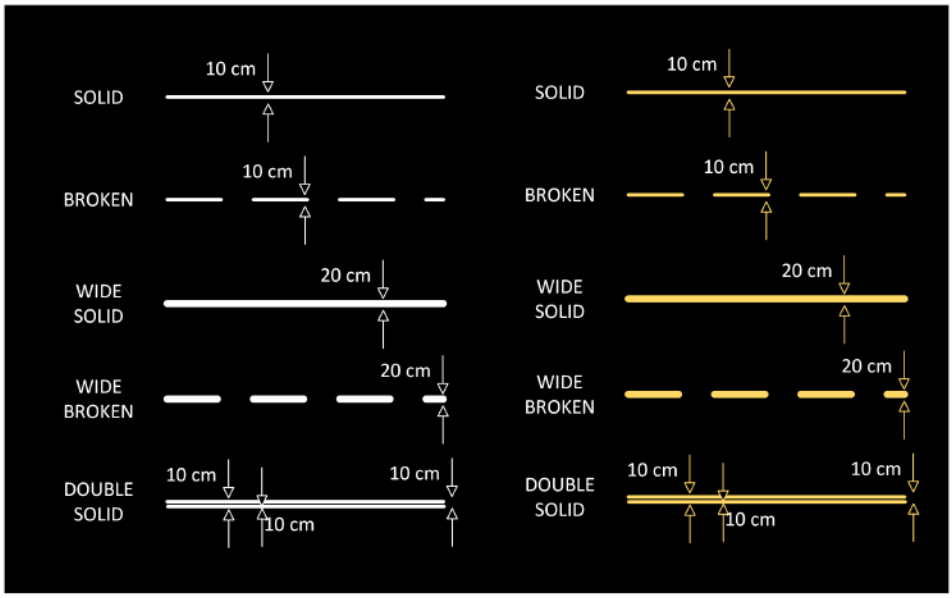


Figure 7.2: Longitudinal line types of interest in the Glover Road Area.

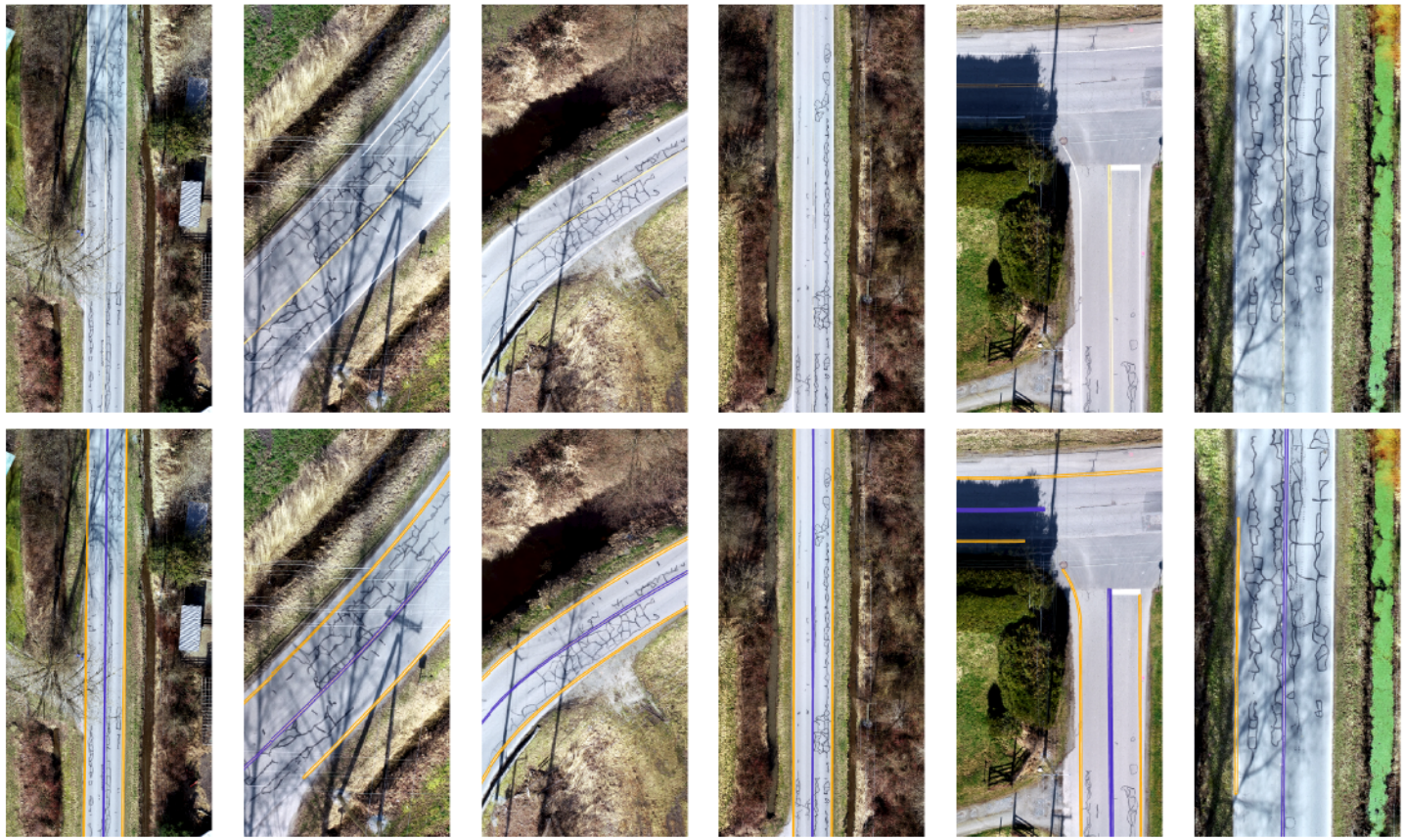


Figure 7.3: Examples of the Glover Road dataset. Top row: RGB image. Bottom row: RGB image annotation with center-line (purple) and edge-line (yellow) classes.

Drone weight	1391 g
Max Ascent Speed	6 m/s (automatic flight); 5 m/s (manual control)
Max Descent Speed	3 m/s
Max Speed	31 mph (50 kph)(P-mode)
Max Flight Time	Approx. 30 minutes
Gimbal Stabilization	3-axis (tilt, roll, yaw)
Pitch	-90° to +30°
Max Controllable Angular Speed	90° /s
Angular Vibration Range	0.02°
GNSS Module	GPS GLONASS Galileo
Mapping Accuracy	Mapping accuracy meets the requirements of the ASPRS Accuracy Standards for Digital Orthophotos Class *III*
Ground Sample Distance(GSD)	(H/36.5) cm/pixel where H is the aircraft altitude relative to shooting scene (unit: m)
Data Acquisition Efficiency	Max operating area of approx. 1 km^2 for a single flight(at an altitude of 182 m, i.e., GSD is approx. 5 cm/pixel, meeting the requirements of the ASPRS Accuracy Standards for Digital Orthophotos Class III

Table 7.1: Drone hardware specifications during Glover Road Mapping Mission.

Classes and data preparation

The dataset considers two classes that describe commonly used traffic painted lines on the road: center-line and edgeline. The image annotation was carried out manually by experts with precision to obtain reliable ground truth data, as shown in Fig. 7.3. Free-drawn polygons were used to label the pixels in the image. Annotations are composed of n polygons of land categories inside a given grid cell. We assigned a class to the pixels within each polygon, either center-line or edge-line. Then, all the polygons are converted to raster. It should be noted that the annotation of the data that occurred in the LRS1 dataset is for training and testing of the models. The LRS2 dataset was used only for testing experiments with the already trained models, This means that the data being evaluated has never seen before. Likewise, in the labelling tasks we focus on labelling representative data of the lines painted on rural roads. In order to achieve a greater generalization performance, various road sections were

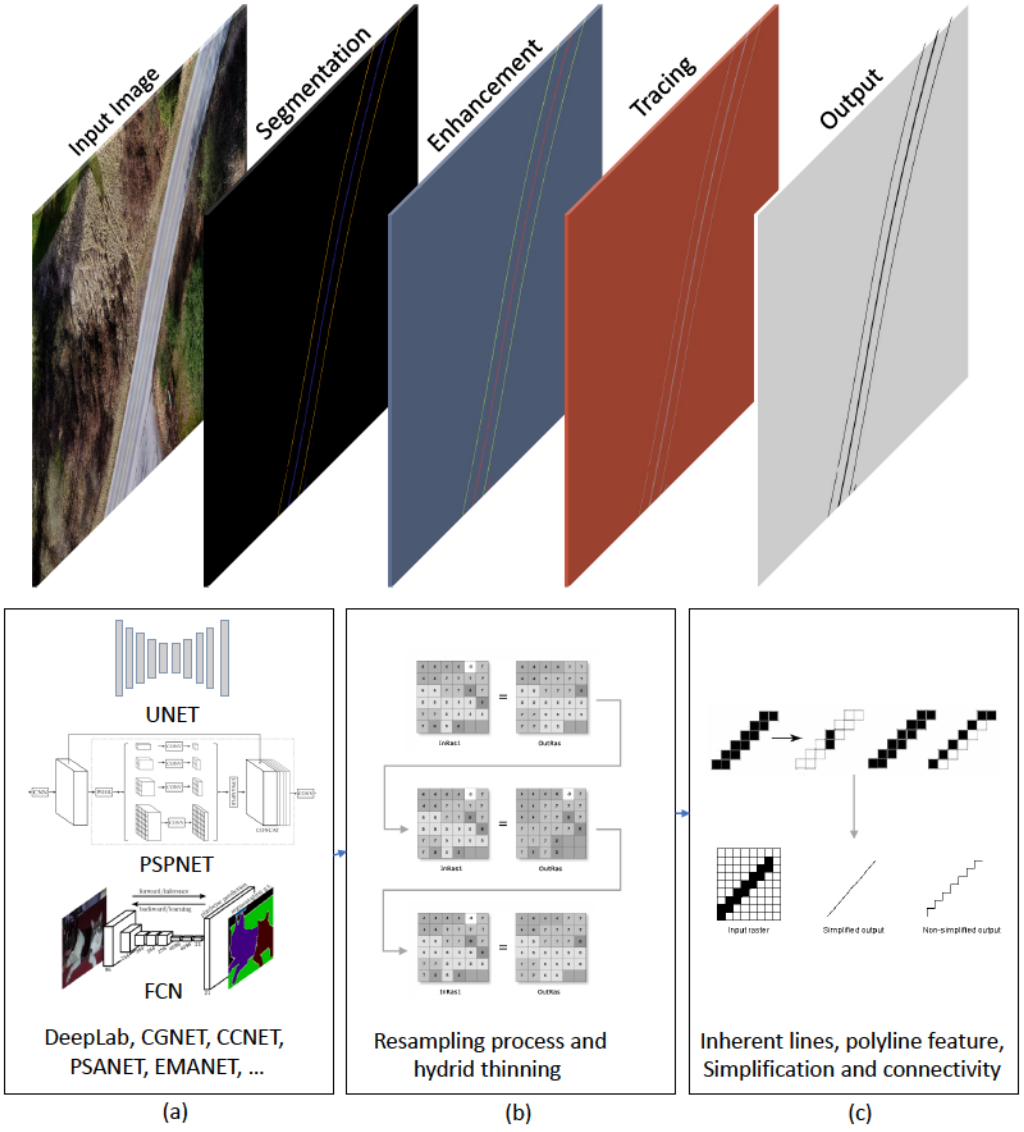


Figure 7.4: Proposed framework for automatic road marking survey.

chosen to obtain a diverse variability of data. The criteria for variability include the orientation of the lines (whether vertical or horizontal), conditions of the lines, areas with curves in the road, shadows that cause changes in intensity, occlusions of trees or power cables, different conditions of pavements, among others. While augmentation techniques can help in theory, when working with large-scale data, it is better to leverage real data and assess the outcome before conducting augmentation. After data labelling, a thorough visual inspection of the ground truth data was performed to confirm that the annotations were correct for further use. The labelled data in LRS1 was cropped and stored into a dataset of 1444 images and corresponding label mask with size 256×256. All the images and label masks were divided into 70% training set and 30% test set.

7.3 Methodology

We propose a Deep Learning-based framework for center-line and edge-line detection for road surveying from high-resolution drone imagery. Our framework consists of three consecutive stages: 1) semantic segmentation, 2) enhancing predicted pixels, and 3) lines connection, as shown in Fig. 7.4. First, the input aerial image is passed through a series of semantic segmentation algorithms to obtain initial predicted pixel masks. We compared different SOTA semantic segmentation algorithms to achieve this purpose. The models were selected based on their performances in different applications. Second, the predicted masks are post-processed through a series of filters to remove noise pixels in different region of the images and maintain only important pixel regions. Third, from the masks output from the second stage, neighborhood pixels are associated and evaluated trace center-lines and edge-lines of the road using the whole content of the aerial image.

Initial semantic segmentation stage

Since our main objective is surveying lines, existing segmentation architectures were tested for the initial segmentation. From the resulting masks, the next step is the generation and plotting of useful lines for road marking surveying. These architectures are able to provide a mask with segmented pixels but are insufficient to retrieve suitable lines for survey applications. For this reason, we introduce our framework, which allows us to extract fine center-lines and edge-lines accurately using aerial images. The input aerial image $\mathcal{I}$ with size $D \times D$ is cropped by a set of patches of size $d \times d$ with corresponding ground truth patches of the same size. In order to obtain a better understanding of the algorithm performance, we evaluate patches with $d =$

128 and 256 respectively. Note that patches with intersection-over-union less than 0.7 with respect to the ground truth are discarded so that our dataset maintains class balance before training the models. The resulting patches are the input data in our segmentation stage. We describe below the commonly adopted neural network architectures evaluated in this work. These networks are selected based on their good performances in other applications.

U-Net[110] is a network introduced by Ronneberger for biomedical image segmentation and is composed of an encoder and a decoder. It has been successful in remote sensing applications that require the processing of large scale data. In this sense, the network is capable of producing highly precise semantic segmentation masks in high resolution images. The segmented output is achieved by using a large number of skip connections layers, and more specifically, the encoder stage extracts feature representations of the image at multiple levels. The decoder then projects the discriminative lower resolution features learnt by the encoder onto the pixel space domain, and finally produces a segmented mask. U-Net has blocks of repeated 3×3 convolutions followed by a ReLU (Rectified Linear Unit) activation function and a 2×2 max pooling layer with stride 2 for downsampling. The reverse happens at the decoder, where a 2×2 up-convolution takes place that reduces the number of channels by 2. Additionally, the output at each level of the encoder block is concatenated with the decoder block. The implemented U-Net architecture utilizes ResNet-34 and ResNet-50 models pretrained with the with ImageNet[193] database.

PSPNet Pyramid Scene Parsing Network[194] was introduced for semantic segmentation as U-Net, and it consists of for an encoder and decoder as well. The PSPNet encoder contains the CNN backbone with dilated convolutions (which helps in increasing the receptive field) along with the pyramid pooling module. The pyramid pooling contributes capturing the global context semantically which helps it to classify the pixels based on the global information present in the image. After the encoder has extracted the features of the image, the decoder generates predictions bases on these features. Similar to U-Net, we evaluated PSPNet embedded with ResNet-34 and ResNet-50.

DeepLab[195] semantic image segmentation model uses a fully convolutional neural network (FCN) to predict a dense classification map for each pixel in an image. The model employs atrous convolution, also known as dilated convolution, to increase the receptive field of the network without increasing the number of parameters. Additionally, DeepLab uses an encoder-decoder architecture with skip connections to maintain high-resolution features from earlier layers while also capturing context from later layers. The model has achieved SOTA performance on various benchmarks, including

the PASCAL VOC and Cityscapes datasets, and has been widely adopted for various computer vision applications.

CGNet[196] (Conjugate Gradient-based network) is a deep learning architecture designed for efficient semantic segmentation of high-resolution images. It uses a conjugate gradient-based optimizer to train the network, which reduces memory requirement and speeds up the training process. The architecture is composed of an encoder and decoder network, connected by a bottleneck layer. The encoder network consists of multiple convolutional layers, which progressively reduce the spatial resolution of the input image while increasing the number of channels. The decoder network, on the other hand, consists of upsampling layers and skip connections that progressively restore the spatial resolution of the feature map. In order to reduce the computational complexity of the network, CGNet uses dilated convolutional layers with a large dilation rate, which increases the receptive field of the network without increasing the number of parameters. Additionally, the network uses a depthwise separable convolution, which factorizes a standard convolution into a depthwise convolution followed by a pointwise convolution. This technique further reduces the number of parameters and computation required by the network.

CCNet[197] Criss-Cross Network is a deep learning architecture designed for accurate semantic segmentation of high-resolution images. It uses a novel criss-cross attention module to capture long-range dependencies between different regions of the image, improving the quality of the segmentation results. CCNet consists of an encoder network, a criss-cross attention module, and a decoder network. The encoder network consists of multiple convolutional layers that extract features from the input image. The criss-cross attention module is inserted between the encoder and decoder networks, and it uses a criss-cross pattern to compute attention maps that capture dependencies between different regions of the image. The decoder network then uses these attention maps to generate the final segmentation mask. CCNet also incorporates other advanced techniques, such as dilated convolutional layers and batch normalization, to improve the performance and efficiency of the network. It achieves SOTA results on several semantic segmentation benchmarks, such as Cityscapes, COCO, and ADE20K. Its use of criss-cross attention and other advanced techniques makes it a powerful tool for accurate and efficient semantic segmentation of high-resolution images.

APCNet[198] Adaptive Pyramid Context Network is a deep learning architecture designed for semantic segmentation tasks in computer vision. It uses a multi-scale feature pyramid approach to capture contextual information at different levels of abstraction and adaptively adjusts the receptive field size of convolutional filters to

improve accuracy and efficiency. APCNet has achieved SOTA performance on several benchmark datasets, including Cityscapes and COCO-Stuff.

ANN[199] Artificial Neural Network is a machine learning model inspired by the structure and function of biological neurons in the human brain. It consists of interconnected layers of nodes that perform mathematical operations on input data to produce output predictions. ANN can be used for a wide range of tasks, including classification, regression, and pattern recognition. The effectiveness of ANN depends on the architecture of the network, the choice of activation functions, and the optimization algorithm used to train the model.

DMNet[200] Dual-channel Multi-scale Network is a deep learning architecture designed for image dehazing, which is the task of removing haze and improving visibility in outdoor images. DMNet uses a dual-channel approach to capture both global and local information in the image and multi-scale features to handle different levels of haze density. The global channel captures the overall scene information, while the local channel focuses on the details of the image. DMNet also includes a novel fusion module that combines the information from both channels to produce a clear image. DMNet has achieved SOTA performance on several benchmark datasets, including NTIRE 2018 and RESIDE.

All the above methods allow segmenting images through mask representations to produce resulting pixel classes. However, they differ from each other in terms of the number of parameters, local features, context-aware modules, pooling grids, computational efficiency and memory consumption. In this scenario, we quantitatively and qualitatively evaluate all the mentioned methods in small patches of the proposed dataset and select the best one for road marking extraction tasks.

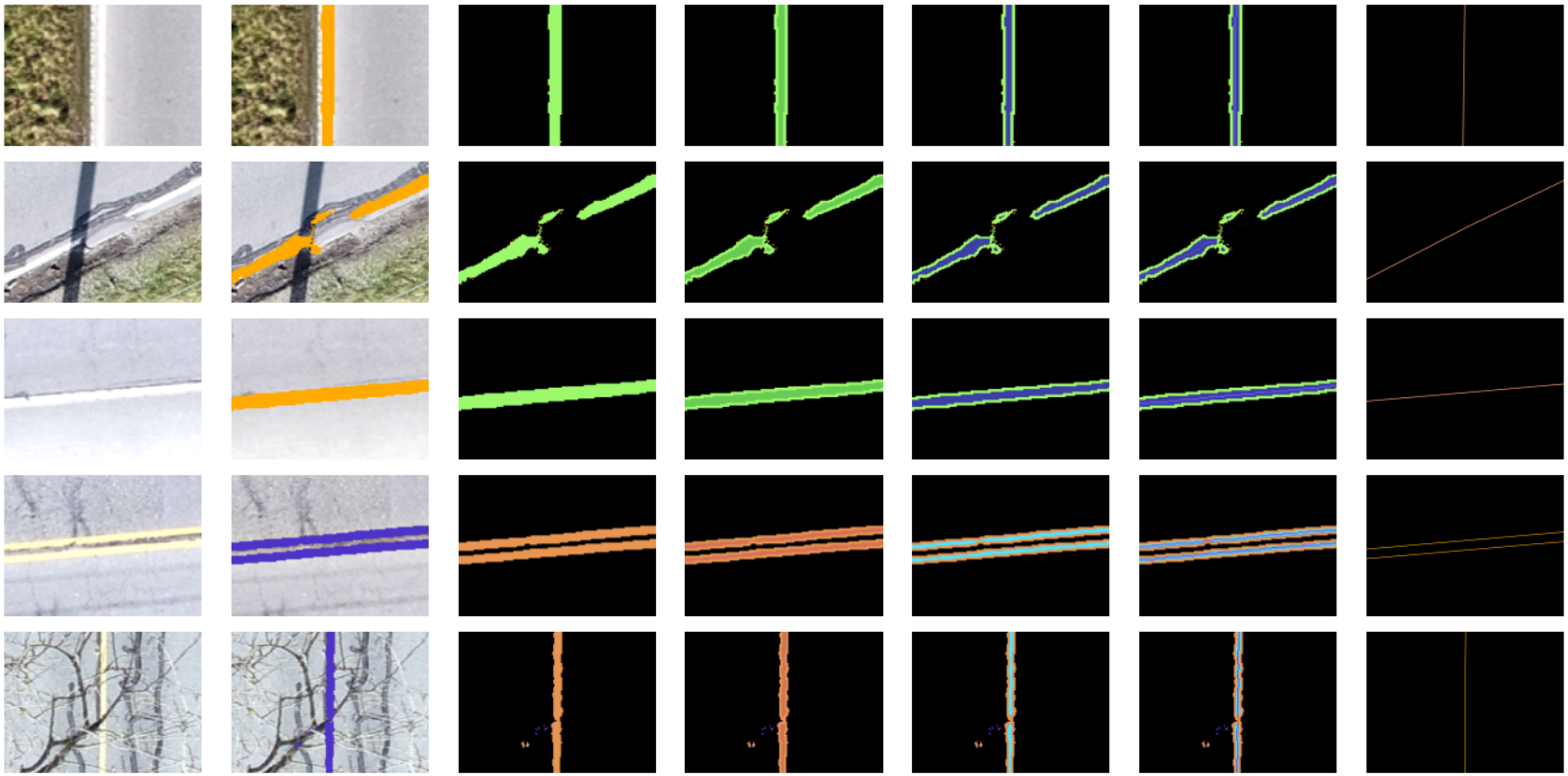


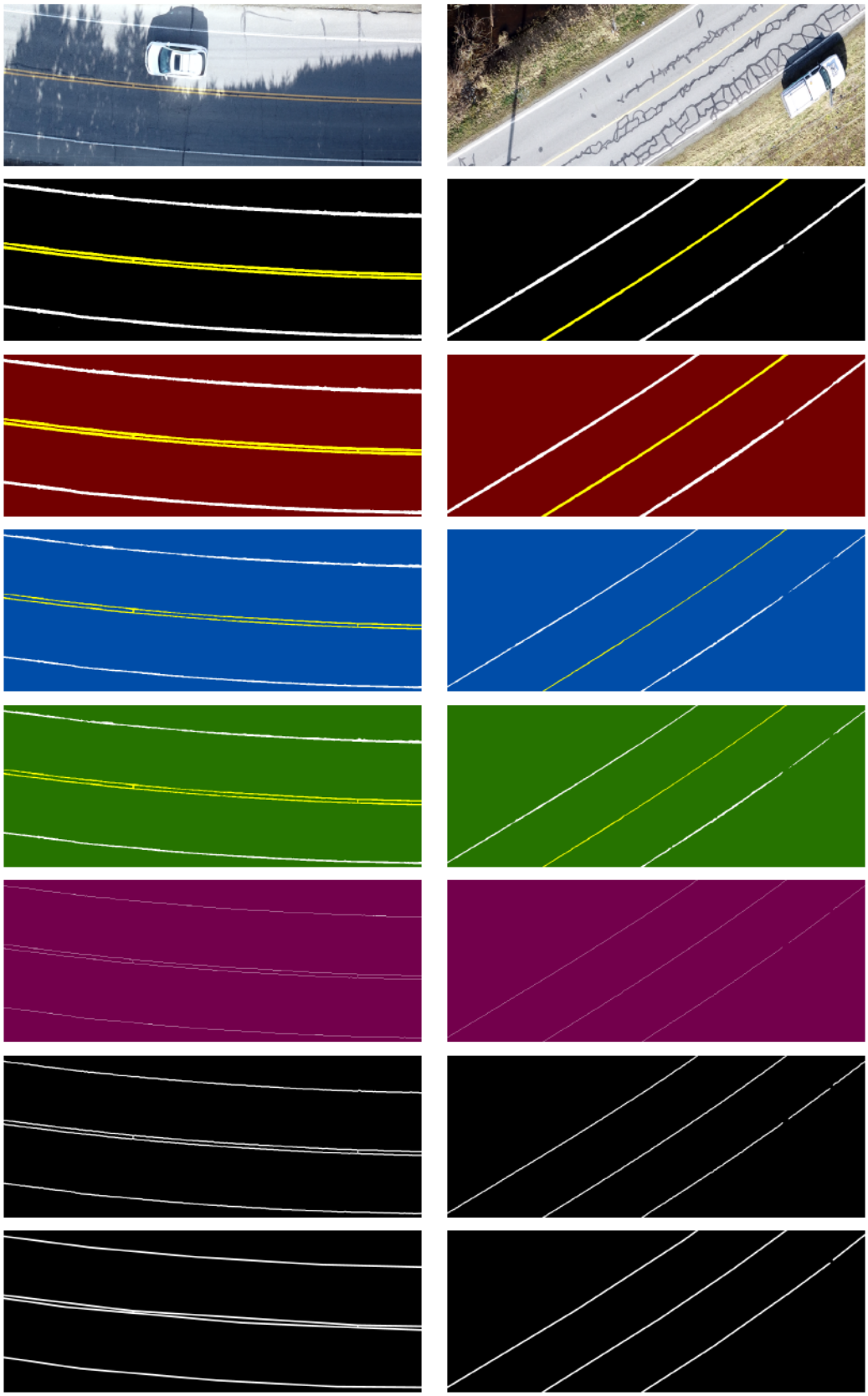
Figure 7.5: Qualitative results in our line generation stage. From top to bottom, a sequence of filters are applied until a final 2D line is extracted.

7.3.1 Pixel regions to polylines

After semantic segmentation of $\mathcal{I}$, the predicted mask $\mathcal{I}_s$ is stored in continuous raster format for further processing. A raster consists of a matrix of cells (or pixels) organized into rows and columns (or a grid) where each cell contains a value representing information. This format also allows access to other layers with geographic information and spectral data of the area. Post-processing techniques are executed at this point to enhance $\mathcal{I}_s$. The procedures allow for highlighting features and reduce the amount of noise mixed in the data. In order to convert the pixel regions to 2D lines at multiple scales, we design a module which employs convolution in cascade to capture a more realistic center-line and edge-lines pixel class and remove counterfeit regions in $\mathcal{I}_s$.

In our proposed module, the first convolution is applied on $\mathcal{I}_s$ using a majority filter. The majority filter induces corner smoothness of rectangular regions [201], allowing for a reduction of the initial complexity of the lines. The majority filter can be expressed mathematically as $F(i,j) = modef(x,y) : (x,y) \in S(i,j)$, where $F(i,j)$ is the filtered output at pixel (i,j), $f(x,y)$ is the intensity value of the pixel at position (x,y) in the input image, $S(i,j)$ is the local neighborhood centered around pixel (i,j), and *mode* represents the mode of the set of intensity values within the neighborhood $S(i,j)$. In simpler terms, the majority filter replaces the intensity value of each pixel in $\mathcal{I}_s$ with the most frequently occurring intensity value within a specified neighborhood around that pixel. This helps to smooth out the image and reduce noise. At this stage, the resulting raster mask $\mathcal{I}_{sf}$ has less noise, and we can focus mainly on processing the target road markings along the image. As previously mentioned, center-lines and egde-lines can be affected by external factors that deteriorate them. These effects are also demonstrated in the spectral data of the images and therefore in the resulting segmentation masks. Therefore, our next step is to shrink the pixel areas of the center-lines and edge-lines in $\mathcal{I}_{sf}$, taking into account that this shrinkage is done towards the center of its extension and not towards its edges. This is because when surveyors survey the road, they take the center of the mark line as a reference so that maps can be generated accurately. Therefore, the following filter refers to the shrinking of $\mathcal{I}_{sf}$.

To achieve this process, we converted our raster $\mathcal{I}_{sf}$ to a binary format, where each pixel is assigned a binary value of either 0 or 1 based on a threshold value. Given the binary raster dataset $\mathcal{I}_b$ with pixel values of either 0 or 1, where 0 represents the background and 1 represents the line features. We apply a shrink operation to this dataset using a structuring element $\mathbf{S}$. The steps for applying the shrink algorithm to



(a) (b)

Figure 7.6: Results of our proposed vectorized lines extraction algorithm.

the binary raster dataset $\mathcal{I}_b$ are as follows. We define a structuring element $\mathbf{S}$, which is a binary matrix representing the neighborhood around each pixel that will be used to probe the image. The structuring element $\mathbf{S}$ can be defined as a matrix of 1s of a specified size, such as a 3x3 matrix or a 5x5 matrix. Let $\mathbf{S}_{ij}$ denote the value of the structuring element at position (i, j). Then, we create a new binary raster dataset, $\hat{\mathcal{I}}_b$, with the same extent, cell size, and projection as the input raster dataset. Initialize all pixels in $\hat{\mathcal{I}}_b$ to 0. A threshold is determined by examining the values in the input raster dataset and selecting a threshold value that separates the line features from the background. The, repeat the following steps until no more changes occur in the output binary raster dataset, $\hat{\mathcal{I}}_b$: a) For each pixel in the input binary raster dataset, $\mathcal{I}_b$. If the pixel value is 0 (background), set the corresponding pixel in the output binary raster dataset, $\hat{\mathcal{I}}_b$, to 0. If the pixel value is 1 (line), probe the neighborhood around the pixel using the structuring element $\mathbf{S}$. If the structuring element $\mathbf{S}$ is completely contained within the line pixels in the neighborhood, set the corresponding pixel in the output binary raster dataset, $\hat{\mathcal{I}}_b$, to 1. Otherwise, set it to 0. The process can be represented by $\hat{\mathcal{I}}_{bij} = \begin{cases} 0 & \text{if } \mathcal{I}_{bij} = 0 \\ 1 & \text{if } \mathcal{I}_{bij} = 1 \text{ and } \sum_{k,l} \mathbf{S}_{kl} \cdot \mathcal{I}_{bi+k-1,j+l-1} = \sum_{k,l} \mathbf{S}_{kl} \end{cases}$ where $\hat{\mathcal{I}}_{bij}$ is the value of the pixel at position (i, j) in the output binary raster dataset, $\mathcal{I}_{bij}$ is the value of the pixel at position (i, j) in the input binary raster dataset, and $\sum_{k,l} \mathbf{S}_{kl} \cdot \mathcal{I}_{bi+k-1,j+l-1}$ is the sum of the products of the corresponding elements in $\mathbf{S}$ and $\mathcal{I}_b$ within the neighborhood around pixel (i, j). Then, b) Replace the input binary raster dataset, $\mathcal{I}_b$, with the output binary raster dataset, $\hat{\mathcal{I}}_b$. Save the final output binary raster dataset, $\mathcal{I}_b$. The algorithm is repeated until convergence, which occurs when the output image no longer changes. The convergence condition ensures that the overall shape and connectivity of the objects in the image are preserved. Note that the choice of structuring element can have a significant impact on the output of the shrink algorithm. A larger structuring element will remove more pixels from the boundaries of objects, resulting in greater shrinkage. Conversely, a smaller structuring element will remove fewer pixels, resulting in less shrinkage. We applied again a convolution filter in $\hat{\mathcal{I}}_b$ to remove noise, and finally convert from raster to vectors lines. We added a thinning module for the conversion of raster lines into vectors (skeletonization) following the proposed method in [202]. Let $\hat{\mathcal{I}}_{\mathbf{b}}$ be the input binary image and $\hat{\mathcal{I}}_{\mathbf{J}}$ be the output thinned image. Let $\hat{\mathcal{I}}_{\mathbf{b}}(x, y)$ and $\hat{\mathcal{I}}_{\mathbf{J}}(x, y)$ denote the pixel values at position (x, y) in the input and output images, respectively. The thinning algorithm can be described by the following steps. Initialize the output image as $\hat{\mathcal{I}}_{\mathbf{J}}(x, y) = \hat{\mathcal{I}}_{\mathbf{b}}(x, y)$ for all (x, y). Then, repeat the following steps until convergence. Traverse the boundary of the binary image: Let $N(x, y)$ be the 3×3

neighborhood centered at pixel (x, y). For each pixel (x, y) in the image, compute:

$A(x, y) = 2 \cdot \hat{\mathcal{G}}_{\mathbf{b}}(x, y) - \hat{\mathcal{G}}_{\mathbf{b}}(x-1, y) - \hat{\mathcal{G}}_{\mathbf{b}}(x+1, y)$

$B(x, y) = 2 \cdot \hat{\mathcal{G}}_{\mathbf{b}}(x, y) - \hat{\mathcal{G}}_{\mathbf{b}}(x, y-1) - \hat{\mathcal{G}}_{\mathbf{b}}(x, y+1)$

$P1(x, y) = \hat{\mathcal{G}}_{\mathbf{b}}(x-1, y) \cdot \hat{\mathcal{G}}_{\mathbf{b}}(x, y) \cdot \hat{\mathcal{G}}_{\mathbf{b}}(x+1, y)$

$P2(x, y) = \hat{\mathcal{G}}_{\mathbf{b}}(x, y-1) \cdot \hat{\mathcal{G}}_{\mathbf{b}}(x, y) \cdot \hat{\mathcal{G}}_{\mathbf{b}}(x, y+1)$

If $A(x, y) == 1$ and $B(x, y) == 0$ and $P1(x, y) == 0$, mark $\hat{\mathcal{G}}_{\mathbf{J}}(x, y)$ for removal.

If $A(x, y) == 0$ and $B(x, y) == 1$ and $P2(x, y) == 0$, mark $\hat{\mathcal{G}}_{\mathbf{J}}(x, y)$ for removal.

Then, remove the marked pixels from the image denoted by $\hat{\mathcal{G}}_{\mathbf{J}}(x, y) = \hat{\mathcal{G}}_{\mathbf{J}}(x, y) - \hat{\mathcal{G}_{\mathbf{M}}}(x, y)$ for all (x, y) in the image, where $\hat{\mathcal{G}_{\mathbf{M}}}(x, y)$ is a binary mask that marks the pixels for removal. Repeat steps (a) and (b) for all eight possible rotations of the image. The process is repeated until convergence, which occurs when no further pixels can be removed from the image. In the above equations, $A(x, y)$ and $B(x, y)$ represent the two conditions for removing a pixel from the image, based on the connectivity of the pixels in the 3×3 neighborhood around each pixel. $P1(x, y)$ and $P2(x, y)$ are additional conditions that help to preserve the topology and connectivity of the image. Note that the thinning algorithm is sensitive to the initial conditions and may produce different results for different starting configurations. In this way, a line of N pixels wide is converted to a line of 1 pixel wide, preserving its original shape.

Fig. 7.5 and Fig. 7.6, show examples of how the algorithm extracts vectorized lines. While the majority filters help reducing inconsistencies due to path decay, our skeletonization module provides a compact form of the target lines. The concatenate modules collaborate reducing the dimensionality of the segmented masks but preserving its topology. Skeletonization provides useful information for segmentation, recognition, and registration, among others [203]. Later, we convert our resulted raster $\hat{\mathcal{G}_{\mathbf{M}}}(x, y)$ to a 1-cell-wide raster line. This format is used to extract the points (x_k, y_k) which are at a distance defined by R. Using the Bresenham algorithm [204], the distance is calculated by $d_{lower} - d_{upper} = 2m(x_k + 1) - 2y_k + 2b - 1$ to determine the points that meet the distance R criteria, with $m = \Delta y / \Delta x$. Thus, the sequence of lines forming the original skeletonization topology is formed. Once the vectorized lines are created, it is important to reduce their complexity, using algorithms such as the conservation of critical points. This algorithm eliminates unnecessary points for the primordial form of the line [205], eliminating abrupt changes in its trajectory. Finally, the vectorized lines are ready for visualization in Computer-aided Design software.

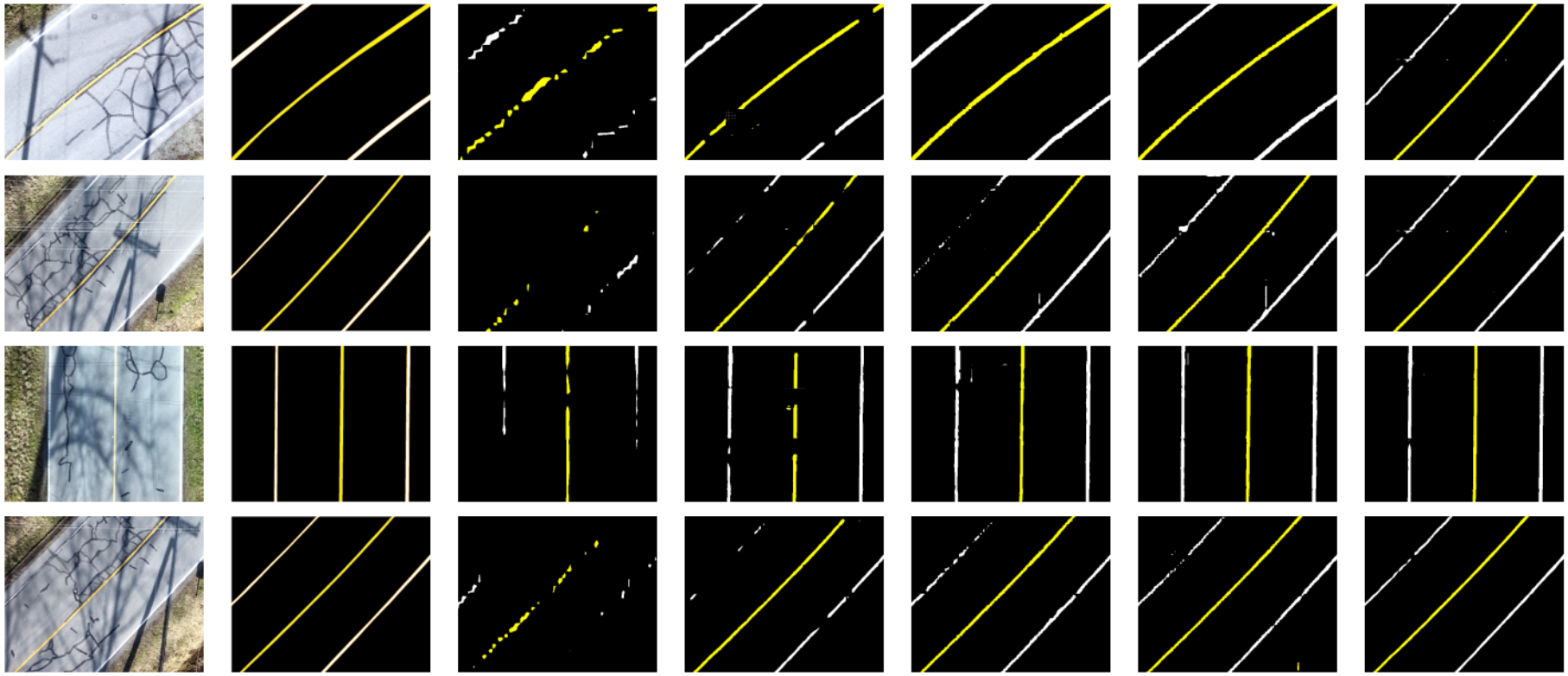


Figure 7.7: Qualitative results examples on the test set for different semantic segmentation models.

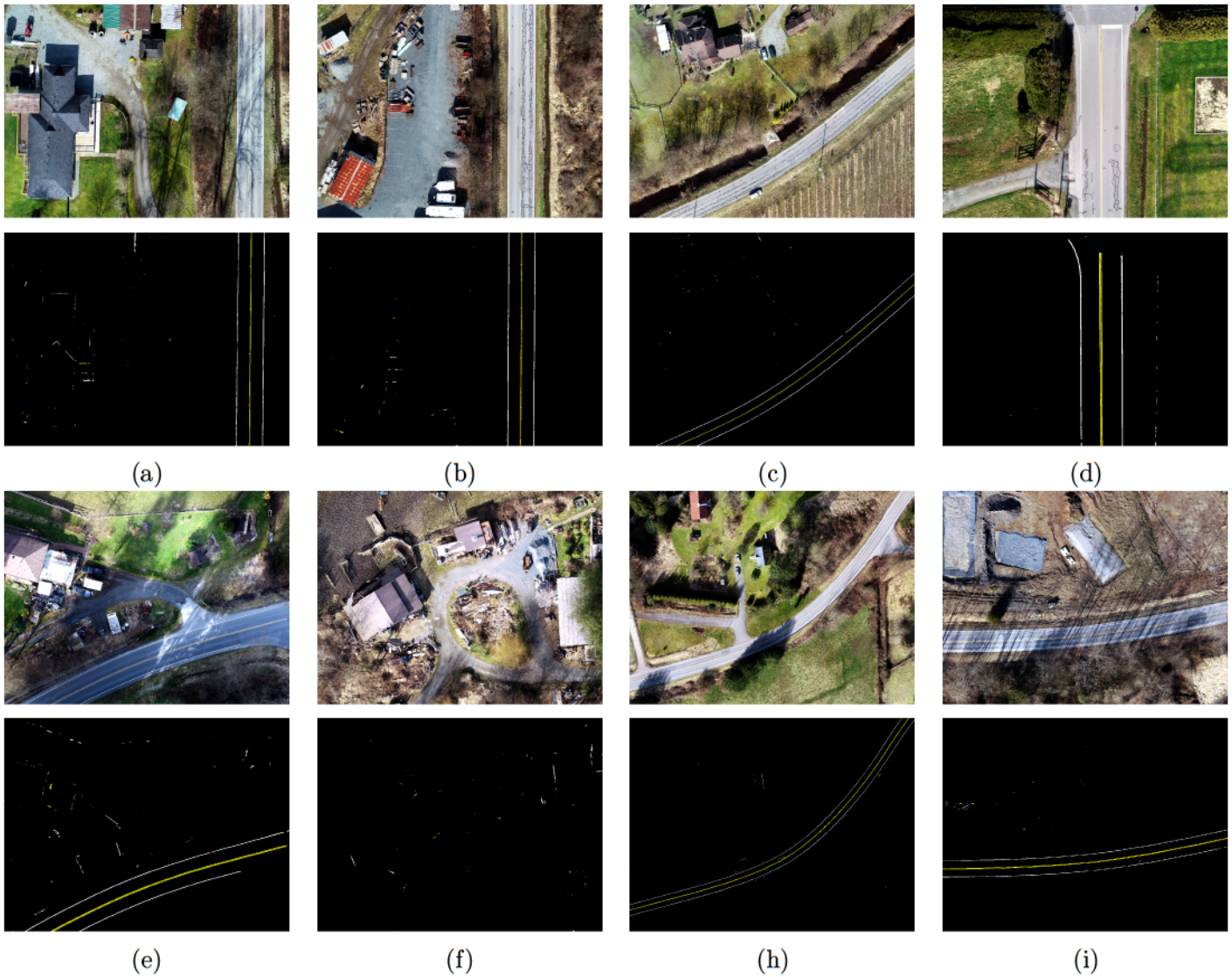


Figure 7.8: Predicted masks using trained U-Net model on Glover Road Dataset.

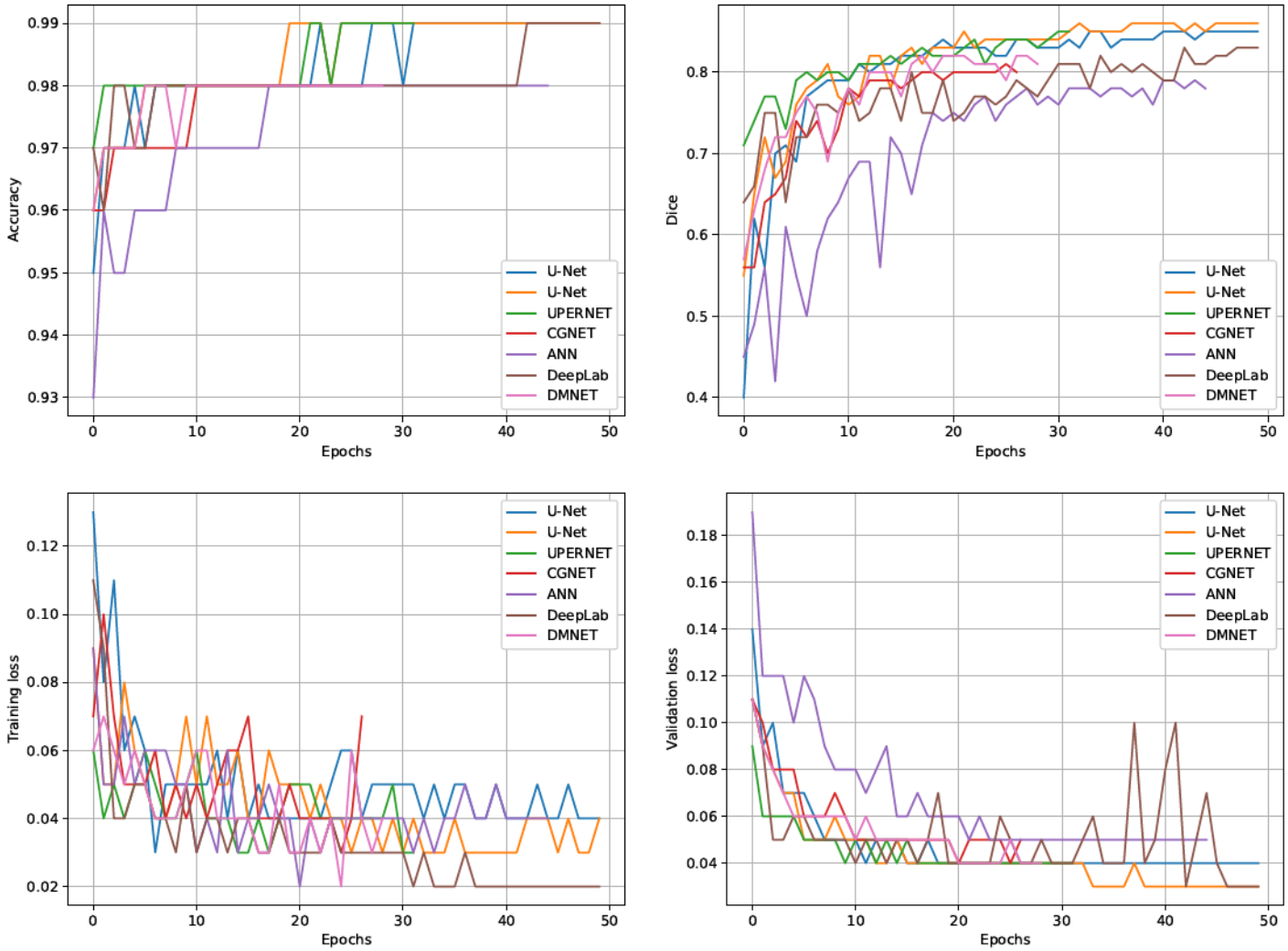


Figure 7.9: Accuracy, Dice, and Training Loss, and Validation Loss results of proposed semantic segmentation methods.

Table 7.2: Quantitative Results for Glover Road with image size 256×256

Model	Backbone	Image Size	Layers	Precision	Recall	F1-Score
U-Net	ResNet-34	256×256	[1,2,3,6]	0.89623	0.92382	0.90957
U-Net	ResNet-50	256×256	[1,2,3,6]	0.89661	0.91693	0.90657
PSPNet	ResNet-34	256×256	[1,2,3,6]	0.90518	0.92255	0.91370
DeepLab	ResNet-34	256×256	[1,2,3,6]	0.87560	0.91985	0.89644
PSPNet	VGG-16	256×256	[1,2,3,6]	0.87329	0.92305	0.89700
U-Net	ResNet-34	256×256	[1,3,6]	0.88726	**0.92853**	0.90710
U-Net	ResNet-34	256×256	[1,2,4,8]	0.88878	0.88547	0.88623
U-Net	ResNet-34	256×256	[1,2,3,6]	0.89080	0.90022	0.89542
PSPNet	ResNet-34	256×256	[1,3,6]	0.89880	0.92225	0.91027
PSPNet	ResNet-50	256×256	[1,3,6]	**0.91263**	0.91719	**0.91489**
PSPNet	VGG-16	256×256	[1,3,6]	0.90377	0.92047	0.91199
PSPNet	VGG-16BN	256×256	[1,3,6]	0.90907	0.91676	0.91285
DeepLab	ResNet-34	256×256	[1,3,6]	0.87468	0.90942	0.89125
DeepLab	ResNet-50	256×256	[1,3,6]	0.88792	0.90975	0.89851
DeepLab	VGG-16	256×256	[1,3,6]	0.64707	0.41563	0.45411
DeepLab	VGG-16BN	256×256	[1,3,6]	0.73292	0.59792	0.64573

Table 7.3: Quantitative Results for Glover Road with image size 128×128

Model	Backbone	Image Size	Layers	Precision	Recall	F1-Score
U-Net	ResNet-34	128×128	[1,2,3,6]	0.93152	0.92358	0.92752
U-Net	ResNet-50	128×128	[1,2,3,6]	0.93868	0.92549	**0.93199**
U-Net	ResNet-34	128×128	[1,3,6]	0.93171	0.91981	0.92569
U-Net	ResNet-50	128×128	[1,3,6]	0.93627	0.92628	0.93121
PSPNet	ResNet-34	128×128	[1,2,3,6]	0.94443	0.89356	0.91778
PSPNet	ResNet-50	128×128	[1,2,3,6]	0.92029	0.89502	0.90691
PSPNet	VGG-16	128×128	[1,2,3,6]	0.94309	0.91165	0.92689
PSPNet	VGG-16BN	128×128	[1,2,3,6]	**0.94585**	0.83539	0.88436
PSPNet	DENSENET121	128×128	[1,2,3,6]	0.89407	0.88579	0.88903
PSPNet	ResNet-34	128×128	[1,3,6]	0.92353	0.89895	0.91092
PSPNet	ResNet-50	128×128	[1,3,6]	0.94215	0.88470	0.91188
PSPNet	DENSENET121	128×128	[1,3,6]	0.89739	0.75605	0.81531
PSPNet	VGG-16BN	128×128	[1,3,6]	0.93295	0.88124	0.90514
DeepLab	ResNet-34	128×128	[1,2,3,6]	0.87253	0.92132	0.89572
DeepLab	VGG-16	128×128	[1,2,3,6]	0.71851	0.56309	0.61787
CGNet		128×128		0.90037	0.91485	0.90749
CCNet		128×128		0.90409	0.89693	0.90048
APCNet		128×128		0.88789	0.92262	0.90430
ANN		128×128		0.89233	0.90994	0.90086
DMNet		128×128		0.89488	0.93368	0.91344
DNLNet		128×128		0.88441	0.92747	0.90506
EMANet		128×128		0.89655	0.91724	0.90649
FCN [206]		128×128		0.90034	**0.93377**	0.91653
GCNet		128×128		0.88264	0.92951	0.90468
HRNet		128×128		0.92351	0.91863	0.92106
OCRNet		128×128		0.91109	0.92073	0.91569
PSANet		128×128		0.86922	0.86558	0.86731
SEM_FPN		128×128		0.92532	0.90796	0.91647
UPERNet		128×128		0.93612	0.91071	0.92310

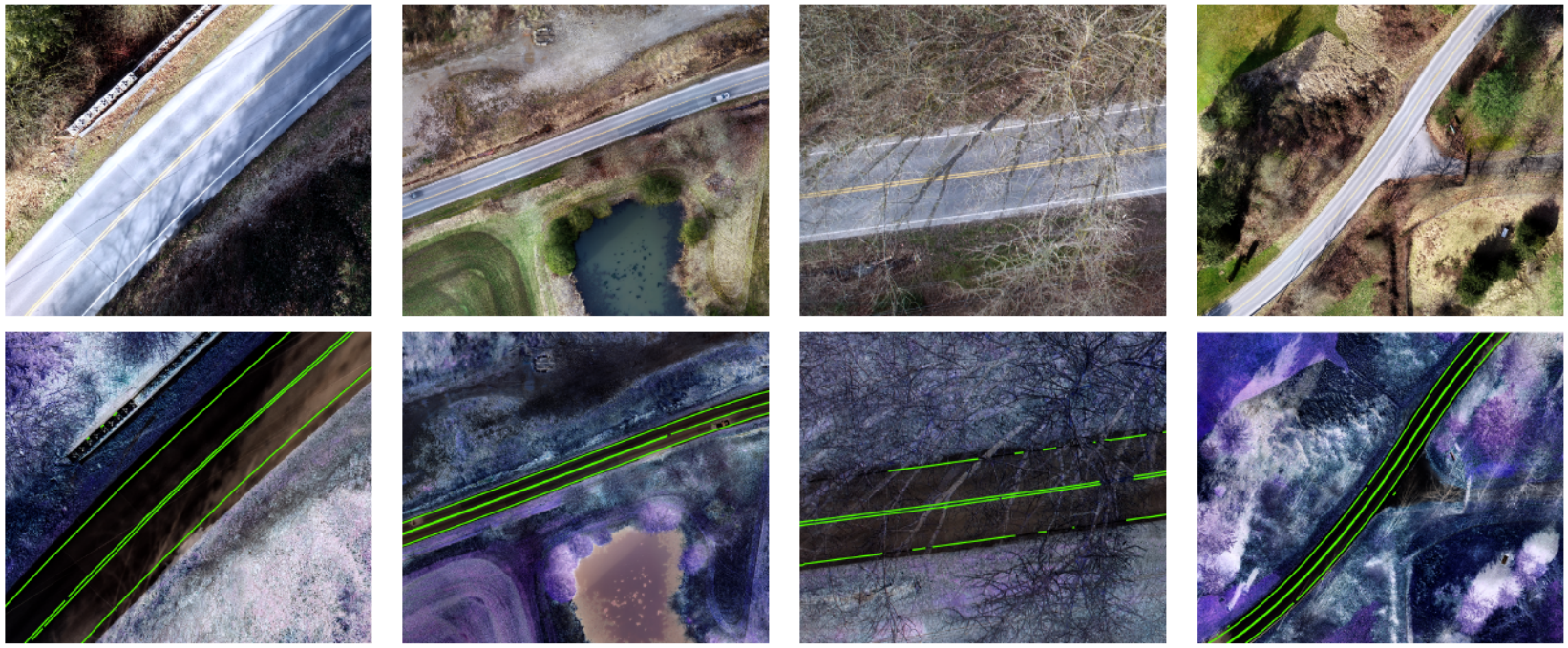


Figure 7.10: Resulted lines (Row 2) using the proposed framework given the input images (Row 1), respectively.

7.3.2 Training and configuration details

In the model training process for semantic segmentation, models were trained from scratch using preconfigured backbones such as ResNet-34, ResNet-50, VGG-16, and VGG-16 with batch normalization. All models were trained with a maximum of 50 epochs with the training performance evaluated after each iteration in case additional training was required. Other fixed values for training include a batch size of 4, input image sizes ($256 \times 256 \times 3$ and $128 \times 128 \times 3$), and a learning rate 1e-5. In order to compare the effectiveness of the models, we use precision, recall, and F1 score as metrics. For training and evaluation, we divided the workload of the different architectures in a Linux system and a Windows machine. Both workstations have a NVIDIA GeForce RTX 2080 SUPER, processor Intel Core i7-10700KF CPU with 3.80GHz×16, and 64GB of RAM.

7.4 Experiment and analysis

7.4.1 Experimental results

In this section, we analyze and compare the performance of different models and backbones in the application of semantic segmentation of painted traffic lines on rural roads using aerial images. We also discuss our post-processing stage, where predicted output masks are refined to project the labelled pixels to 2D lines. We conduct quantitative and qualitative (Fig. 7.7 and Fig. 7.8) experiments for training and evaluation in the Glover Road database (LRS1 area) without data augmentation. Only previous manually extracted data are used to verify the effectiveness of the models against raw data. Table 7.2 and Table 7.3 show the results on the Glover Road Dataset along with different model architectures and backbones. The accuracy, dice, training Loss, and validation loss results of the proposed semantic segmentation methods are shown in Fig. 7.9. The diversity of configuration in this dataset allows us to validate the performance of the model against different scene conditions. PSPNet with VGG-16BN obtained the highest precision (94.45%) compared to the others using input images with size 128x128. PSPNet maintains similar values using backbones Resnet-34, VGG-16, and ResNet-50, with 94.44%, 94.30%, and 94.21%, respectively. Regarding precision, the previous models are followed by U-Net Resnet-50 and UperNet with 93.86% and 91.61%. This demonstrates the performance of these models in predicting positive pixels in classification tasks but not how false negatives can affect them. The CCNet, CGNet, and FCN models maintained precision values around 90% compared to the previous methods. Note that the PSPNet

with VGG-16BN with an input image size of 256x256, the precision dropped off by around 4% (90.90%) which shows how the performance can be affected when input data is at various scales. DeepLab (with VGG-16) was the model that obtained the lowest precision for both data input sizes, with 71.85% and 64.70%, respectively. Regarding sensitivity (Recall), FCN and DMNet reached 93.37%, followed by CGNet with 92.95%. U-Net with ResNet-34 and ResNet-50 backbones is also one of the best for recall at above 92%. We selected the F1-score metric as the best to measure the performance of all models since it uses both precision and recall in its measurement. U-Net (with ResNet-50) proved to be the model with the best prediction in the segmentation of traffic lines, with a score of 93.19%. Even U-Net, using fewer convolution layers, performs at 93.12% with the same data (image size 128x128), followed by PSPNet with VGG-16 (92.68%). FCN, SEM-FPN, OCRNet, and DMNet also demonstrated acceptable performance, with recall values around 91%. We noticed many of the algorithms can fail or ignore the classification of some pixels in regions of our dataset. Even in more challenging scenarios as shown in Fig. 7.8, U-Net tends to misclassify pixels in areas that have similar patterns to the lines marked on the pavement. For example, in Fig. 7.8(a) the input image and predicted mask of the pavement lines can be observed with great precision. However, it is also observed that there are false classifications on and around the roof of the house, and near a garage at the top of the image. Likewise, Fig. 7.8(b-c) and Fig. 7.8(f) a greater misclassification is notable, because there are several objects in the area that share similar color intensities as the pavement lines, and especially in patterns that contain edges. In Fig. 7.8(h), a greater coverage can be seen in the segmentation of the image, and this is due to the fact that the area has mostly pixels of vegetation and land. Therefore, it can be concluded that these architectures can be largely affected by the context of the image. However, our framework overcomes this problem by using our cascading modules to mitigate these artifacts that affect line extraction. In our stage of refinement and generation of lines, the convolutions applied in each step allowed us to mitigate the noise associated with the segmentation stage. We experimented with various parameters that allowed us to generate quite precise lines.

7.5 Conclusion

Considering the manual and labor-intensive road marking tasks in the land surveying and engineering industries, we propose an automatic framework for road marking surveying of center-line and edge-line using high resolution aerial images. We construct our self-made benchmark dataset of a rural scene including annotated images

and geospatial metadata of linear features. Our framework first performs an initial segmentation of the imagery data using SOTA algorithms for semantic segmentation, which provides masks of target traffic lines on the road for the center-line and edge-line classes. We compare the selected models with multiple configurations either in the input data or model parameters to check the segmentation performance in challenging cases. Then, the predicted masks are processed in a refinement stage to extract thin and smooth connected lines without losing the pattern of their shape. The refinement stage makes use of majority filters at different scales to generalize and reduce single pixel misclassification, then the processed masks go through a dedicated thinning stage and generate the skeletonization of the lines. Our quantitative and qualitative experiments indicated that our proposed framework obtained satisfactory results and can be an alternative for surveyors. This work demonstrates the feasibility of autonomous survey tasks using high resolution drone imagery, which can be used for pavement marking surveys over a wide rural area for rapid assessment. In the future, we will focus on improving processing time as well as evaluating other pavement marking lines.

Chapter 8

Conclusion and Future work

In this thesis, we explored pattern recognition scenarios involving various sensors modalities, where existing methodologies were insufficient to effectively tackle the computational and pattern-related challenges, primarily due to scalability demands, constrained contextual information, or the presence of dynamic environmental factors. In this thesis we suggest potential future directions which interested researchers can continue to explore in local-area and remote-sensing fields.

For local-area, in-situ object detection and pose estimation, we propose a semi-supervised strategy to obtain the pose of texture-less objects placed in a complex industrial scene. Our system is trained on synthetic images using the solid model of the target object. We locate the RoI that delimits the object's structure through an unsupervised segmentation method which does not require learning stage reducing computation. Then, a learning-based algorithm is trained to recognize objects in different perspectives using HOG features and invariant moments. The estimation of the object's pose is based on a matching technique using a DTW method to compare vectors of different sizes without losing representative characteristics of the features. Our pipeline demonstrates better detection and recognition accuracy compared with existing methods on texture-less objects. In future work, we will explore optimization to further improve computational time and accuracy at different stages.

In video analysis for local-area, in-situ traffic surveys, this study presents a robust and adaptable multi-class traffic analysis system capable of accurate detection, classification, and tracking in diverse urban environments. The results demonstrate its effectiveness under varying conditions, making it a valuable asset in traffic analysis and management. Future research could focus on optimizing the system for real-time applications, enhancing accuracy under challenging conditions, integrating with smart city infrastructure for urban management, addressing privacy concerns, and developing user-friendly interfaces. It could also investigate more active learn-

ing techniques to prioritize samples that are diverse or dissimilar from the existing labelled data. Collaborative efforts between researchers, urban planners, and government agencies can further advance the field, ensuring practical applications and continuous improvements in traffic analysis systems.

In UAV feature extraction for WAM, we showed the advantage of CNN-based supervised learning with signal processing to address the problem of linear feature extraction from aerial photography for 3D footprint generation. We showed a method that uses a CNN-based semantic segmentation model to extract interest feature lines in the pixel domain and proposed an essential stage for refining and extruding the pixels to 3D lines. However, a major limitation of this work is the demanding computational time for high-resolution images with thousands of pixels. The future directions of this research involve investigating the possibility of processing patches or tiles simultaneously using parallel processing techniques. Another processing strategy can be based on the level of detail (LOD). Closer objects may require higher-resolution tiles, while distant objects can be represented with lower-resolution tiles. LOD techniques help optimize processing power and memory usage, ensuring that resources are allocated where they are most needed.

In LiDAR feature extraction for WAM, we introduced a new method and dataset suitable for automatic land survey from TLS-LiDAR data. Our CNN-based method addressed the point resolution variations and adaptability to different shapes and sizes of objects, and the lack of training data. However, a major limitation of the current study is the demanding computational time for dense point clouds with billions of points. Future directions of this research involve investigate the possibility of using multi-GPU clusters to accelerate pre-processing that currently works on only a single GPU. Another potential direction is employing the use of AR/VR technology, on the premise that the immersive experience might expedite and enhance attribute annotation. Also, we will conduct research to render our urban features assets into a simulation platform ,e.g., NVIDIA Omniverse or the Microsoft Mesh network, for use in 3D visualization, and possibly for use as an asset management solution based on the blockchain paradigm such as Metaverse.

As we reflect on this research, determining the most efficient approach for different data dimensions poses a nuanced challenge, often leading to a reliance on learning-based algorithms for their superior performance. However, the scarcity of data can affect learning-based approaches, necessitating a careful consideration of factors such as input data format, dataset availability, complexity, expected output, inference time, and memory usage. Object recognition further emphasizes factors like robustness against size variation, data resolution, occlusion, shape variation, texture, and

color. Despite these considerations, no single architecture universally accommodates all data dimensions due to the inherent complexity of diverse input data structures. While certain 2D/3D fields offer alternatives such as synthetic data generation from CAD models, the realism and variability of such approaches may constrain their performance. Addressing the localization of multifold object sizes with 3D detector algorithms is feasible, but the trade-off involves adopting dense representative discretization and incurring higher computational costs for real-time applications.

Finally, this thesis discloses innovative solutions to intricate pattern recognition challenges across diverse sensor modalities, showing advancements in object detection, pose estimation, traffic analysis, and feature extraction from aerial imagery and LiDAR data. The results demonstrate the effectiveness of the proposed methodologies, offering valuable insights for both current and future research in the fields of local-area and remote-sensing contexts. As we reflect on the findings, the identified limitations and future directions underscore the evolving nature of the field, inviting researchers to build upon these contributions, refine existing techniques, and explore new horizons in the domain of pattern recognition. This work not only contributes to the academic discourse but also lays the foundation for continued progress and innovation in the broader domain of spatial and geospatial data analysis.

Related Journals and Conferences

- **Lugo, Gabriel**, Yair Andrade-Ambriz, Siddharth Jha, Dora Almanza-Ojeda, and Irene Cheng. *"Automatic Pavement Marking Survey from High-Resolution Drone Imagery using a Multiscale Semantic Segmentation Framework."* IEEE Transactions on Geoscience and Remote Sensing. (submitted)

- **Lugo, Gabriel**, Joey Quinlan, Lingrui Zhou, Md Nahid Sadik and Irene Cheng. *"Active Learning for Multi-Class Vehicle Categorization and Traffic Analysis in complex environments."* In 2023 IEEE International Symposium on Multimedia.

- **Lugo, Gabriel**, Rutvik Chauhan and Irene Cheng. *"Exploring terrestrial point clouds with Google Street View for discovery and fine-grained catalog of urban objects."* In 2023 IEEE International Symposium on Multimedia.

- **Lugo, Gabriel**, Amit Upreti, and Irene Cheng. *"Multiscale point feature object localization for hydrant surveying using LiDAR."* In 2022 IEEE 5th International Conference on Multimedia Information Processing and Retrieval (MIPR), pp. 372-378. IEEE, 2022.

- **Lugo, Gabriel**, Ryan Li, Rutvik Chauhan, Zihao Wang, Palak Tiwary, Utkarsh Pandey, Archi Patel, Steve Rombough, Rod Schatz, and Irene Cheng. *"LiSurveying: A high-resolution TLS-LiDAR benchmark."* Computers & Graphics 107 (2022): 116-130.

- **Lugo, Gabriel**, Nasim Hajari, and Irene Cheng. *"Semi-supervised learning approach for localization and pose estimation of texture-less objects in cluttered scenes."* Array (2022): 100247.

- Hajari, Nasim, **Gabriel Lugo Bustillo**, Harsh Sharma, and Irene Cheng. *"Marker-less 3d object recognition and 6D pose estimation for homogeneous textureless objects: An rgb-d approach."* Sensors 20, no. 18 (2020): 5098.

- **Lugo, Gabriel**, Nasim Hajari, Ashley Reddy, and Irene Cheng. *"Textureless object recognition using an RGB-D sensor."* In Smart Multimedia: Second International Conference, ICSM 2019, San Diego, CA, USA, December 16–18, 2019, Revised Selected Papers 2, pp. 13-27. Springer International Publishing, 2020.

Along with our academic research and publications, our final aim is to employ our algorithms to resolve practical challenges within real-world pattern recognition

processing pipelines, ultimately enhancing efficiency and accelerating processes. In collaboration with the Advanced Research and Innovation (ART) program at McElhanney, we set up our proposed algorithms within McElhanney production pipelines for algorithm validation and operationalization depending on the data and sensor modality. The amount of data generated by real-world LiDAR can exceed 25 gigabytes per site, depending on the type of LiDAR scanner, area of study, and sensor parameters during acquisition. Video recording exceeds 20 GB per camera location for a 24 hour period. An aerial image provided by the company is normally 2 to 4 gigabytes, depending the coverage area. As researchers, we typically begin with a subset of data to explore, get insights, conduct proof-of-concept and validate our methods. Later, we must adapt the proposed algorithms to integrate with McElhanney's processing pipelines and align with their specific requirements. This process is the culmination of months of meticulous data preparation, annotation, and intensive training using the comprehensive datasets provided by McElhanney. Our algorithms have been refined and honed in the Multimedia Research Lab (MRC) at the University of Alberta. Its outputs are verified and evaluated by geospatial operators before becoming a standard processing module for McElhanney. Through this collaborative effort, we are not only advancing the frontiers of scientific knowledge but also delivering practical, cutting-edge solutions that drive innovation and efficiency in local-area and wide-area remote sensing.

Bibliography

[1] N. Hajari, G. Lugo Bustillo, H. Sharma, and I. Cheng, "Marker-less 3d object recognition and 6d pose estimation for homogeneous textureless objects: An rgb-d approach," *Sensors*, vol. 20, no. 18, p. 5098, 2020.

[2] S. Hinterstoisser *et al.*, "Multimodal templates for real-time detection of textureless objects in heavily cluttered scenes," in *2011 international conference on computer vision*, IEEE, 2011, pp. 858–865.

[3] S. Hinterstoisser *et al.*, "Gradient response maps for real-time detection of textureless objects," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 5, pp. 876–888, 2011.

[4] J. Hexner and R. R. Hagege, "2d-3d pose estimation of heterogeneous objects using a region based approach," *International Journal of Computer Vision*, vol. 118, no. 1, pp. 95–112, 2016.

[5] E. Brachmann, F. Michel, A. Krull, M. Y. Yang, S. Gumhold, *et al.*, "Uncertainty-driven 6d pose estimation of objects and scenes from a single rgb image," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 3364–3372.

[6] H. Tjaden, U. Schwanecke, and E. Schomer, "Real-time monocular pose estimation of 3d objects using temporally consistent local color histograms," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 124–132.

[7] M. Liang and X. Hu, "Recurrent convolutional neural network for object recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3367–3375.

[8] 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*, 2014, pp. 580–587.

[9] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448.

[10] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 6, pp. 1137–1149, 2016.

[11] 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*, 2016, pp. 770–778.

[12] C. Szegedy *et al.*, "Going deeper with convolutions," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1–9.

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

[14] T. T. Pham, T.-T. Do, N. Sünderhauf, and I. Reid, "Scenecut: Joint geometric and object segmentation for indoor scenes," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2018, pp. 3213–3220.

[15] S. Gupta, R. Girshick, P. Arbeláez, and J. Malik, "Learning rich features from rgb-d images for object detection and segmentation," in *European conference on computer vision*, Springer, 2014, pp. 345–360.

[16] S. Gupta, P. Arbelaez, and J. Malik, "Perceptual organization and recognition of indoor scenes from rgb-d images," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2013, pp. 564–571.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 142–158, 2015.

[18] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Region-based convolutional networks for accurate object detection and segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 1, pp. 142–158, 2015.

[19] A. Mousavian, D. Anguelov, J. Flynn, and J. Kosecka, "3d bounding box estimation using deep learning and geometry," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 7074–7082.

[20] J. Wu *et al.*, "Single image 3d interpreter network," in *European Conference on Computer Vision*, Springer, 2016, pp. 365–382.

[21] S. Mahendran, H. Ali, and R. Vidal, "3d pose regression using convolutional neural networks," in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, 2017, pp. 2174–2182.

[22] G. Pavlakos, X. Zhou, A. Chan, K. G. Derpanis, and K. Daniilidis, "6-dof object pose from semantic keypoints," in *2017 IEEE international conference on robotics and automation (ICRA)*, IEEE, 2017, pp. 2011–2018.

[23] M. Rad and V. Lepetit, "Bb8: A scalable, accurate, robust to partial occlusion method for predicting the 3d poses of challenging objects without using depth," in *Proceedings of the IEEE International Conference on Computer Vision*, 2017, pp. 3828–3836.

[24] X. Deng, A. Mousavian, Y. Xiang, F. Xia, T. Bretl, and D. Fox, "Poserbpf: A rao–blackwellized particle filter for 6-d object pose tracking," *IEEE Transactions on Robotics*, vol. 37, no. 5, pp. 1328–1342, 2021.

[25] W. Ding, X. Wang, H. Liu, and B. Hu, "An empirical study of shape recognition in ensemble learning context," in *2018 International Conference on Wavelet Analysis and Pattern Recognition (ICWAPR)*, IEEE, 2018, pp. 256–261.

[26] J Flusser, "Invariant shape description and measure of object similarity," in *1992 International Conference on Image Processing and its Applications*, IET, 1992, pp. 139–142.

[27] M. E. Celebi and Y. A. Aslandogan, "A comparative study of three moment-based shape descriptors," in *International Conference on Information Technology: Coding and Computing (ITCC'05)-Volume II*, IEEE, vol. 1, 2005, pp. 788–793.

[28] E. Davies, "Machine vision: Theory, algorithms, practicalities. academic press., 2-nd edition," 1997.

[29] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE transactions on information theory*, vol. 8, no. 2, pp. 179–187, 1962.

[30] D. Guru and H. Nagendraswamy, "Symbolic representation of two-dimensional shapes," *Pattern Recognition Letters*, vol. 28, no. 1, pp. 144–155, 2007.

[31] S.-M. Lee, A. L. Abbott, N. A. Clark, and P. A. Araman, "A shape representation for planar curves by shape signature harmonic embedding," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, IEEE, vol. 2, 2006, pp. 1940–1947.

[32] D. Zhang and G. Lu, "A comparative study of curvature scale space and fourier descriptors for shape-based image retrieval," *Journal of Visual Communication and Image Representation*, vol. 14, no. 1, pp. 39–57, 2003.

[33] G.-H. Chuang and C.-C. Kuo, "Wavelet descriptor of planar curves: Theory and applications," *IEEE Transactions on Image Processing*, vol. 5, no. 1, pp. 56–70, 1996.

[34] H. Freeman, "On the encoding of arbitrary geometric configurations," *IRE Transactions on Electronic Computers*, no. 2, pp. 260–268, 1961.

[35] K. Kpalma and J. Ronsin, "Multiscale contour description for pattern recognition," *Pattern Recognition Letters*, vol. 27, no. 13, pp. 1545–1559, 2006.

[36] Y. W. Chen and Y. Q. Chen, "Invariant description and retrieval of planar shapes using radon composite features," *IEEE Transactions on Signal Processing*, vol. 56, no. 10, pp. 4762–4771, 2008.

[37] M. Sonka, V. Hlavac, and R. Boyle, *Image processing, analysis, and machine vision*. Cengage Learning, 2014.

[38] S. Abbasi, F. Mokhtarian, and J. Kittler, "Enhancing css-based shape retrieval for objects with shallow concavities," *Image and vision computing*, vol. 18, no. 3, pp. 199–211, 2000.

[39] M. Yang, K. Kpalma, and J. Ronsin, "Scale-controlled area difference shape descriptor," in *Document Recognition and Retrieval XIV*, International Society for Optics and Photonics, vol. 6500, 2007, p. 650 003.

[40] C. Huang and J. Huang, "A fast hog descriptor using lookup table and integral image," *arXiv preprint arXiv:1703.06256*, 2017.

[41] H. Cevikalp and B. Triggs, "Efficient object detection using cascades of nearest convex model classifiers," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, IEEE, 2012, pp. 3138–3145.

[42] M. A. Rahim, M. S. Azam, N. Hossain, and M. R. Islam, "Face recognition using local binary patterns (lbp)," *Global Journal of Computer Science and Technology*, 2013.

[43] Y. Feng, X. An, and X. Liu, "The application of scale invariant feature transform fused with shape model in the human face recognition," in *Advanced Information Management, Communicates, Electronic and Automation Control Conference (IMCEC), 2016 IEEE*, IEEE, 2016, pp. 1716–1720.

[44] M. Calonder, V. Lepetit, C. Strecha, and P. Fua, "Brief: Binary robust independent elementary features," in *European conference on computer vision*, Springer, 2010, pp. 778–792.

[45] R. Verma and R. Kaur, "Enhanced character recognition using surf feature and neural network technique," *IJCSIT) International Journal of Computer Science and Information Technologies*, vol. 5, no. 4, pp. 5565–5570, 2014.

[46] E. Rublee, V. Rabaud, K. Konolige, and G. R. Bradski, "Orb: An efficient alternative to sift or surf.," in *ICCV*, Citeseer, vol. 11, 2011, p. 2.

[47] Y. Kortli, M. Jridi, A. Al Falou, and M. Atri, "A comparative study of cfs, lbp, hog, sift, surf, and brief for security and face recognition,"

[48] Z. Ren, S. Gao, L.-T. Chia, and I. W.-H. Tsang, "Region-based saliency detection and its application in object recognition," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, no. 5, pp. 769–779, 2013.

[49] M. A. Hearst, S. T. Dumais, E. Osuna, J. Platt, and B. Scholkopf, "Support vector machines," *IEEE Intelligent Systems and their applications*, vol. 13, no. 4, pp. 18–28, 1998.

[50] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[51] S. Balakrishnama and A. Ganapathiraju, "Linear discriminant analysis-a brief tutorial," *Institute for Signal and information Processing*, vol. 18, pp. 1–8, 1998.

[52] N. Friedman, D. Geiger, and M. Goldszmidt, "Bayesian network classifiers," *Machine learning*, vol. 29, no. 2-3, pp. 131–163, 1997.

[53] L. Breiman, "Random forests," *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001.

[54] T. Kohonen, "Learning vector quantization," in *Self-organizing maps*, Springer, 1995, pp. 175–189.

[55] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[56] A. K. Jain and R. C. Dubes, "Algorithms for clustering data," 1988.

[59] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[58] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[60] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018.

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

[62] G. Jocher, *Ultralytics/yolov5: Yolov5 in pytorch*, 2020. [Online]. Available: https://github.com/ultralytics/yolov5.

[63] C. Li *et al.*, "Yolov6: A single-stage object detection framework for industrial applications," *arXiv preprint arXiv:2209.02976*, 2022.

[64] G. Jocher, *Ultralytics yolov8 docs*, 2023. [Online]. Available: https://docs.ultralytics.com/.

[65] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*, Springer, 2020, pp. 213–229.

[66] W. Lv *et al.*, "Detrs beat yolos on real-time object detection," *arXiv preprint arXiv:2304.08069*, 2023.

[67] G. Welch, G. Bishop, *et al.*, "An introduction to the kalman filter," 1995.

[68] W. Feng, D. Ji, Y. Wang, S. Chang, H. Ren, and W. Gan, "Challenges on large scale surveillance video analysis," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 69–697. DOI: 10.1109/CVPRW.2018.00017.

[69] E. A. Wan and R. Van Der Merwe, "The unscented kalman filter for nonlinear estimation," in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No. 00EX373)*, Ieee, 2000, pp. 153–158.

[70] P. M. Djuric *et al.*, "Particle filtering," *IEEE signal processing magazine*, vol. 20, no. 5, pp. 19–38, 2003.

[71] D. Comaniciu and P. Meer, "Mean shift: A robust approach toward feature space analysis," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 24, no. 5, pp. 603–619, 2002.

[72] J. K. Suhr, "Kanade-lucas-tomasi (klt) feature tracker," *Computer Vision (EEE6503)*, pp. 9–18, 2009.

[73] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1409–1422, 2011.

[74] D. Exner, E. Bruns, D. Kurz, A. Grundhöfer, and O. Bimber, "Fast and robust camshift tracking," in *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition-Workshops*, IEEE, 2010, pp. 9–16.

[75] D. S. Bolme, J. R. Beveridge, B. A. Draper, and Y. M. Lui, "Visual object tracking using adaptive correlation filters," in *2010 IEEE computer society conference on computer vision and pattern recognition*, IEEE, 2010, pp. 2544–2550.

[76] N. M. Al-Shakarji, F. Bunyak, G. Seetharaman, and K. Palaniappan, "Multi-object tracking cascade with multi-step data association and occlusion handling," in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 2018, pp. 1–6. DOI: 10.1109/AVSS.2018.8639321.

[77] R. E. Kalman, "A New Approach to Linear Filtering and Prediction Problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, Mar. 1960. [Online]. Available: https://doi.org/10.1115/1.3662552.

[78] H. W. Kuhn, "The hungarian method for the assignment problem," *Naval Research Logistics Quarterly*, vol. 2, no. 1-2, pp. 83–97, 1955.

[79] N. Wojke, A. Bewley, and D. Paulus, "Simple online and realtime tracking with a deep association metric," in *2017 IEEE international conference on image processing (ICIP)*, IEEE, 2017, pp. 3645–3649.

[80] Y. Du *et al.*, "Strongsort: Make deepsort great again," *IEEE Transactions on Multimedia*, 2023.

[81] P. Voigtlaender *et al.*, "Mots: Multi-object tracking and segmentation," in *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 2019, pp. 7942–7951.

[82] E. Baser, V. Balasubramanian, P. Bhattacharyya, and K. Czarnecki, "Fantrack: 3d multi-object tracking with feature association network," in *2019 IEEE Intelligent Vehicles Symposium (IV)*, IEEE, 2019, pp. 1426–1433.

[83] K. Duan, S. Bai, L. Xie, H. Qi, Q. Huang, and Q. Tian, "Centernet: Keypoint triplets for object detection," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 6569–6578.

[84] B. Mocanu, R. Tapu, and T. Zaharia, "Single object tracking using offline trained deep regression networks," in *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, IEEE, 2017, pp. 1–6.

[85] B. Li, W. Wu, Q. Wang, F. Zhang, J. Xing, and J. Yan, "Siamrpn++: Evolution of siamese visual tracking with very deep networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 4282–4291.

[86] Q. Yu, C. Yang, H. Fan, and H. Wei, "Latent-mvcnn: 3d shape recognition using multiple views from pre-defined or random viewpoints," *Neural Processing Letters*, vol. 52, no. 1, pp. 581–602, 2020.

[87] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao, "Gvcnn: Group-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 264–272.

[88] T. Yu, J. Meng, and J. Yuan, "Multi-view harmonized bilinear network for 3d object recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 186–194.

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

[90] A. X. Chang *et al.*, "Shapenet: An information-rich 3d model repository," *arXiv preprint arXiv:1512.03012*, 2015.

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

[92] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Advances in neural information processing systems*, vol. 30, 2017.

[93] H. Ma, Y. Zhao, and Q. He, "Road extraction from high resolution remote sensing image based on mathematics morphology and seed growth," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Science*, vol. 37, 2008.

[94] D. B. Bong, K. C. Lai, and A. Joseph, "Automatic road network recognition and extraction for urban planning," *International Journal of Applied Science, Engineering and Technology*, vol. 5, no. 1, pp. 209–215, 2009.

[95] Q. Zhang and I. Couloigner, "Accurate centerline detection and line width estimation of thick lines using the radon transform," *IEEE Transactions on image processing*, vol. 16, no. 2, pp. 310–316, 2007.

[96] C. Poullis and S. You, "Delineation and geometric modeling of road networks," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 65, no. 2, pp. 165–181, 2010.

[97] Z. Miao, W. Shi, H. Zhang, and X. Wang, "Road centerline extraction from high-resolution imagery based on shape features and multivariate adaptive regression splines," *IEEE geoscience and remote sensing letters*, vol. 10, no. 3, pp. 583–587, 2012.

[98] J. Yuan, D. Wang, and R. Li, "Remote sensing image segmentation by combining spectral and texture features," *IEEE Transactions on geoscience and remote sensing*, vol. 52, no. 1, pp. 16–24, 2013.

[99] D. Chaudhuri, N. K. Kushwaha, and A. Samal, "Semi-automated road detection from high resolution satellite images by directional morphological enhancement and segmentation techniques," *IEEE journal of selected topics in applied earth observations and remote sensing*, vol. 5, no. 5, pp. 1538–1544, 2012.

[100] R. Liu, J. Song, Q. Miao, P. Xu, and Q. Xue, "Road centerlines extraction from high resolution images based on an improved directional segmentation and road probability," *Neurocomputing*, vol. 212, pp. 88–95, 2016.

[101] P. Anil and S Natarajan, "Automatic road extraction from high resolution imagery based on statistical region merging and skeletonization," *International Journal of Engineering Science and Technology*, vol. 2, no. 3, pp. 165–171, 2010.

[102] J. Cheng, Y. Guan, X. Ku, and J. Sun, "Semi-automatic road centerline extraction in high-resolution sar images based on circular template matching," in *2011 International Conference on Electric Information and Control Engineering*, IEEE, 2011, pp. 1688–1691.

[103] J. Cheng, W. Ding, X. Ku, and J. Sun, "Road extraction from high-resolution sar images via automatic local detecting and human-guided global tracking," *International Journal of Antennas and Propagation*, vol. 2012, 2012.

[104] M Saati, J Amini, and M Maboudi, "A method for automatic road extraction of high resolution sar imagery," *Journal of the Indian Society of Remote Sensing*, vol. 43, pp. 697–707, 2015.

[105] J. Cheng and G. Gao, "Parallel particle filter for tracking road centrelines from high-resolution sar images using detected road junctions as initial seed points," *International Journal of Remote Sensing*, vol. 37, no. 20, pp. 4979–5000, 2016.

[106] S. Valero, J. Chanussot, J. A. Benediktsson, H. Talbot, and B. Waske, "Advanced directional mathematical morphology for the detection of the road network in very high resolution remote sensing images," *Pattern Recognition Letters*, vol. 31, no. 10, pp. 1120–1127, 2010.

[107] Y. Bae, W.-H. Lee, Y. Choi, Y. W. Jeon, and J. B. Ra, "Automatic road extraction from remote sensing images based on a normalized second derivative map," *IEEE Geoscience and remote sensing letters*, vol. 12, no. 9, pp. 1858–1862, 2015.

[108] L. Courtrai and S. Lefèvre, "Morphological path filtering at the region scale for efficient and robust road network extraction from satellite imagery," *Pattern Recognition Letters*, vol. 83, pp. 195–204, 2016.

[109] V. Badrinarayanan, A. Kendall, and R. Cipolla, "Segnet: A deep convolutional encoder-decoder architecture for image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 12, pp. 2481–2495, 2017.

[110] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, Springer, 2015, pp. 234–241.

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

[112] X. Chen, X. Wu, D. K. Prasad, B. Wu, O. Postolache, and Y. Yang, "Pixel-wise ship identification from maritime images via a semantic segmentation model," *IEEE Sensors Journal*, vol. 22, no. 18, pp. 18 180–18 191, 2022.

[113] S. Vachmanus, A. A. Ravankar, T. Emaru, and Y. Kobayashi, "Multi-modal sensor fusion-based semantic segmentation for snow driving scenarios," *IEEE sensors journal*, vol. 21, no. 15, pp. 16 839–16 851, 2021.

[114] J. Billson, M. S. Islam, X. Sun, and I. Cheng, "Water body extraction from sentinel-2 imagery with deep convolutional networks and pixelwise category transplantation," *Remote Sensing*, vol. 15, no. 5, p. 1253, 2023.

[115] G. Cheng, Y. Wang, S. Xu, H. Wang, S. Xiang, and C. Pan, "Automatic road detection and centerline extraction via cascaded end-to-end convolutional neural network," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 55, no. 6, pp. 3322–3337, 2017.

[116] C. Ventura, J. Pont-Tuset, S. Caelles, K.-K. Maninis, and L. Van Gool, "Iterative deep learning for road topology extraction," *arXiv preprint arXiv:1808.09814*, 2018.

[117] X. Lu, Y. Zhong, Z. Zheng, and L. Zhang, "Gamsnet: Globally aware road detection network with multi-scale residual learning," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 175, pp. 340–352, 2021.

[118] J. Mei, R.-J. Li, W. Gao, and M.-M. Cheng, "Coanet: Connectivity attention network for road extraction from satellite imagery," *IEEE Transactions on Image Processing*, vol. 30, pp. 8540–8552, 2021.

[119] X. Lu, Y. Zhong, Z. Zheng, and J. Wang, "Cross-domain road detection based on global-local adversarial learning framework from very high resolution satellite imagery," *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 180, pp. 296–312, 2021.

[120] X. Lu *et al.*, "Cascaded multi-task road extraction network for road surface, centerline, and edge extraction," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 60, pp. 1–14, 2022.

[121] M. Sundermeyer, Z.-C. Marton, M. Durner, M. Brucker, and R. Triebel, "Implicit 3d orientation learning for 6d object detection from rgb images," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 699–715.

[122] X. Peng, "Combine color and shape in real-time detection of texture-less objects," *Computer Vision and Image Understanding*, vol. 135, pp. 31–48, 2015.

[123] A. Crivellaro, M. Rad, Y. Verdie, K. M. Yi, P. Fua, and V. Lepetit, "Robust 3d object tracking from monocular images using stable parts," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1465–1479, 2017.

[124] W. Kehl, F. Manhardt, F. Tombari, S. Ilic, and N. Navab, "Ssd-6d: Making rgb-based 3d detection and 6d pose estimation great again," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 1521–1529.

[125] T. Hodan, P. Haluza, Š. Obdržálek, J. Matas, M. Lourakis, and X. Zabulis, "T-less: An rgb-d dataset for 6d pose estimation of texture-less objects," in *2017 IEEE Winter Conference on Applications of Computer Vision (WACV)*, IEEE, 2017, pp. 880–888.

[126] S. Beucher and F. Meyer, "The morphological approach to segmentation: The watershed transformation," in *Mathematical morphology in image processing*, CRC Press, 2018, pp. 433–481.

[127] R. K. McConnell, *Method of and apparatus for pattern recognition*, US Patent 4,567,610, 1986.

[128] A. C. Ruifrok, D. A. Johnston, *et al.*, "Quantification of histochemical staining by color deconvolution," *Analytical and quantitative cytology and histology*, vol. 23, no. 4, pp. 291–299, 2001.

[129] W. Kehl, F. Milletari, F. Tombari, S. Ilic, and N. Navab, "Deep learning of local rgb-d patches for 3d object detection and 6d pose estimation," in *European conference on computer vision*, Springer, 2016, pp. 205–220.

[130] J. Vidal, C.-Y. Lin, and R. Martí, "6d pose estimation using an improved method based on point pair features," in *2018 4th international conference on control, automation and robotics (iccar)*, IEEE, 2018, pp. 405–409.

[131] B. Tekin, S. N. Sinha, and P. Fua, "Real-time seamless single shot 6d object pose prediction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2018, pp. 292–301.

[132] *Transport Canada, Transportation in Canada, Statistical Addendum 2022*, https://tc.canada.ca/en/corporate-services/transparency/corporate-management-reporting/transportation-canada-annual-reports/transportation-canada-2022, Accessed: 2023-08-24.

[133] *Transport Canada, Transportation in Canada, Statistical Addendum 2021*, https://tc.canada.ca/en/corporate-services/transparency/corporate-management-reporting/transportation-canada-annual-reports/2021/transportation-canada-2021, Accessed: 2023-08-24.

[134] *Transport Canada, Transportation in Canada, Statistical Addendum 2020*, https://tc.canada.ca/en/corporate-services/transparency/corporate-management-reporting/transportation-canada-annual-reports/transportation-canada-2020-overview-report, Accessed: 2023-08-24.

[135] W. Balid, H. Tafish, and H. H. Refai, "Intelligent vehicle counting and classification sensor for real-time traffic surveillance," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 6, pp. 1784–1794, 2017.

[136] T. Celik and H. Kusetogullari, "Solar-powered automated road surveillance system for speed violation detection," *IEEE Transactions on Industrial Electronics*, vol. 57, no. 9, pp. 3216–3227, 2009.

[137] L Lin, X. Han, R Ding, G Li, S. C. Lu, and Q Hong, "A new rechargeable intelligent vehicle detection sensor," in *Journal of Physics: Conference Series*, IOP Publishing, vol. 13, 2005, p. 102.

[138] Y.-K. Ki and D.-K. Baik, "Model for accurate speed measurement using double-loop detectors," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 4, pp. 1094–1101, 2006.

[139] L.-E. Y. Mimbela, L. A. Klein, *et al.*, "Summary of vehicle detection and surveillance technologies used in intelligent transportation systems," 2007.

[140] S. Y. Cheung and P. P. Varaiya, "Traffic surveillance by wireless sensor networks," Ph.D. dissertation, Citeseer, 2006.

[141] S. Gupte, O. Masoud, R. F. Martin, and N. P. Papanikolopoulos, "Detection and classification of vehicles," *IEEE Transactions on intelligent transportation systems*, vol. 3, no. 1, pp. 37–47, 2002.

[142] T. N. Schoepflin and D. J. Dailey, "Dynamic camera calibration of roadside traffic management cameras," in *Proceedings. The IEEE 5th International Conference on Intelligent Transportation Systems*, IEEE, 2002, pp. 25–30.

[143] J. Zhou, D. Gao, and D. Zhang, "Moving vehicle detection for automatic traffic monitoring," *IEEE transactions on vehicular technology*, vol. 56, no. 1, pp. 51–59, 2007.

[144] Y. Wang, "Real-time moving vehicle detection with cast shadow removal in video based on conditional random field," *IEEE transactions on circuits and systems for video technology*, vol. 19, no. 3, pp. 437–441, 2009.

[145] X. Zhang, B. Story, and D. Rajan, "Night time vehicle detection and tracking by fusing vehicle parts from multiple cameras," *IEEE transactions on intelligent transportation systems*, vol. 23, no. 7, pp. 8136–8156, 2021.

[146] M. A. B. Zuraimi and F. H. K. Zaman, "Vehicle detection and tracking using yolo and deepsort," in *2021 IEEE 11th IEEE Symposium on Computer Applications & Industrial Electronics (ISCAIE)*, IEEE, 2021, pp. 23–29.

[147] V. S. Sindhu, "Vehicle identification from traffic video surveillance using yolov4," in *2021 5th international conference on intelligent computing and control systems (ICICCS)*, IEEE, 2021, pp. 1768–1775.

[148] G. Oltean, C. Florea, R. Orghidan, and V. Oltean, "Towards real time vehicle counting using yolo-tiny and fast motion estimation," in *2019 IEEE 25th international symposium for design and technology in electronic packaging (SIITME)*, IEEE, 2019, pp. 240–243.

[149] N. Jain, S. Yerragolla, T. Guha, and Mohana, "Performance analysis of object detection and tracking algorithms for traffic surveillance applications using neural networks," in *2019 Third International conference on I-SMAC (IoT in Social, Mobile, Analytics and Cloud) (I-SMAC)*, 2019, pp. 690–696. DOI: 10.1109/I-SMAC47947.2019.9032502.

[150] O. Sener and S. Savarese, "Active learning for convolutional neural networks: A core-set approach," *arXiv preprint arXiv:1708.00489*, 2017.

[151] L. Li, H. He, and J. Li, "Entropy-based sampling approaches for multi-class imbalanced problems," *IEEE Transactions on Knowledge and Data Engineering*, vol. 32, no. 11, pp. 2159–2170, 2019.

[152] Y. Li, B. Fan, W. Zhang, W. Ding, and J. Yin, "Deep active learning for object detection," *Information Sciences*, vol. 579, pp. 418–433, 2021.

[153] J. Wu, J. Chen, and D. Huang, "Entropy-based active learning for object detection with progressive diversity constraint," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 9397–9406.

[154] J. Alori *et al.*, *Tryolabs/norfair: V2.2.0*, version v2.2.0, Jan. 2023. DOI: 10.5281/zenodo.7504727. [Online]. Available: https://doi.org/10.5281/zenodo.7504727.

[155] H. Song, H. Liang, H. Li, Z. Dai, and X. Yun, "Vision-based vehicle detection and counting system using deep learning in highway scenes," *European Transport Research Review*, vol. 11, no. 1, pp. 1–16, 2019.

[156] Z. Rahman, A. M. Ami, and M. A. Ullah, "A real-time wrong-way vehicle detection based on yolo and centroid tracking," in *2020 IEEE Region 10 Symposium (TENSYMP)*, IEEE, 2020, pp. 916–920.

[157] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, PMLR, 2021, pp. 8748–8763.

[158] T. Hackel, N. Savinov, L. Ladicky, J. D. Wegner, K. Schindler, and M. Pollefeys, "Semantic3d. net: A new large-scale point cloud classification benchmark," *arXiv preprint arXiv:1704.03847*, 2017.

[159] D. Munoz, J. A. Bagnell, N. Vandapel, and M. Hebert, "Contextual classification with functional max-margin markov networks," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, IEEE, 2009, pp. 975–982.

[160] B. Vallet, M. Brédif, A. Serna, B. Marcotegui, and N. Paparoditis, "Terramobilita/iqmulus urban point cloud analysis benchmark," *Computers & Graphics*, vol. 49, pp. 126–133, 2015.

[161] W. Tan *et al.*, "Toronto-3d: A large-scale mobile lidar dataset for semantic segmentation of urban roadways," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 202–203.

[162] X. Roynard, J.-E. Deschaud, and F. Goulette, "Paris-lille-3d: A large and high-quality ground-truth urban point cloud dataset for automatic segmentation and classification," *The International Journal of Robotics Research*, vol. 37, no. 6, pp. 545–557, 2018.

[163] T. Ku *et al.*, "Shrec 2020: 3d point cloud semantic segmentation for street scenes," *Computers & Graphics*, vol. 93, pp. 13–24, 2020.

[164] F. Rottensteiner *et al.*, "The isprs benchmark on urban object classification and 3d building reconstruction," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences I-3 (2012), Nr. 1*, vol. 1, no. 1, pp. 293–298, 2012.

[165] N. M. Singer and V. K. Asari, "Dales objects: A large scale benchmark dataset for instance segmentation in aerial lidar," *IEEE Access*, vol. 9, pp. 97 495–97 504, 2021.

[166] Z. Ye *et al.*, "Lasdu: A large-scale aerial lidar dataset for semantic labeling in dense urban areas," *ISPRS International Journal of Geo-Information*, vol. 9, no. 7, p. 450, 2020.

[167] S. Zolanvari *et al.*, "Dublincity: Annotated lidar point cloud and its applications," *arXiv preprint arXiv:1909.03613*, 2019.

[168] Z. T. A. L. J. Y. Xinke Li Chongshou Li and R. H. Yuwei Wu Jing Tang, "Campus3d: A photogrammetry point cloud benchmark for hierarchical understanding of outdoor scene.," in *Proceedings of the 28th ACM International Conference on Multimedia*, ser. MM '20, Seattle, WA, USA: Association for Computing Machinery, 2020, ISBN: 978-1-4503-7988-5/20/10. DOI: 10.1145/3394171.3413661.

[169] Q. Hu, B. Yang, S. Khalid, W. Xiao, N. Trigoni, and A. Markham, "Towards semantic segmentation of urban-scale 3d point clouds: A dataset, benchmarks and challenges," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 4977–4987.

[170] J. Behley *et al.*, "Semantickitti: A dataset for semantic scene understanding of lidar sequences," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9297–9307.

[171] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1231–1237, 2013.

[172] A. Patil, S. Malla, H. Gang, and Y.-T. Chen, "The h3d dataset for full-surround 3d multi-object detection and tracking in crowded urban scenes," in *2019 International Conference on Robotics and Automation (ICRA)*, IEEE, 2019, pp. 9552–9557.

[173] M.-F. Chang *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8748–8757.

[174] J. Houston *et al.*, "One thousand and one hours: Self-driving motion prediction dataset," *arXiv preprint arXiv:2006.14480*, 2020.

[175] Q.-H. Pham *et al.*, "A* 3d dataset: Towards autonomous driving in challenging environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, 2020, pp. 2267–2273.

[176] P. Sun *et al.*, "Scalability in perception for autonomous driving: Waymo open dataset," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 2446–2454.

[177] H. Caesar *et al.*, "Nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[178] M. E. Atik, Z. Duran, and D. Z. Seker, "Machine learning-based supervised classification of point clouds using multiscale geometric features," *ISPRS International Journal of Geo-Information*, vol. 10, no. 3, p. 187, 2021.

[179] K. Zhang, M. Hao, J. Wang, C. W. de Silva, and C. Fu, "Linked dynamic graph cnn: Learning on point cloud via linking hierarchical features," *arXiv preprint arXiv:1904.10014*, 2019.

[180] Y. Wang, Y. Sun, Z. Liu, S. E. Sarma, M. M. Bronstein, and J. M. Solomon, "Dynamic graph cnn for learning on point clouds," *Acm Transactions On Graphics (tog)*, vol. 38, no. 5, pp. 1–12, 2019.

[181] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, 2017, pp. 1907–1915.

[182] J. Ku, M. Mozifian, J. Lee, A. Harakeh, and S. L. Waslander, "Joint 3d proposal generation and object detection from view aggregation," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, 2018, pp. 1–8.

[183] M. Liang, B. Yang, S. Wang, and R. Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 641–656.

[184] M. Liang, B. Yang, Y. Chen, R. Hu, and R. Urtasun, "Multi-task multi-sensor fusion for 3d object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 7345–7353.

[185] H. Lu, X. Chen, G. Zhang, Q. Zhou, Y. Ma, and Y. Zhao, "Scanet: Spatial-channel attention network for 3d object detection," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, IEEE, 2019, pp. 1992–1996.

[186] W. Niu *et al.*, "Rt3d: Achieving real-time execution of 3d convolutional neural networks on mobile devices," *arXiv preprint arXiv:2007.09835*, 2020.

[187] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "Ipod: Intensive point-based object detector for point cloud," *arXiv preprint arXiv:1812.05276*, 2018.

[188] S. Shi, X. Wang, and H. Li, "Pointrcnn: 3d object proposal generation and detection from point cloud," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2019, pp. 770–779.

[189] J. Zarzar, S. Giancola, and B. Ghanem, "Pointrgcn: Graph convolution networks for 3d vehicles detection refinement," *arXiv preprint arXiv:1911.12236*, 2019.

[190] S. Vora, A. H. Lang, B. Helou, and O. Beijbom, "Pointpainting: Sequential fusion for 3d object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4604–4612.

[191] Z. Yang, Y. Sun, S. Liu, X. Shen, and J. Jia, "Std: Sparse-to-dense 3d object detector for point cloud," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1951–1960.

[192] J. Li and Y. Hu, "A density-aware pointrcnn for 3d objection detection in point clouds," 2020.

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

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

[195] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille, "Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 4, pp. 834–848, 2017.

[196] T. Wu, S. Tang, R. Zhang, J. Cao, and Y. Zhang, "Cgnet: A light-weight context guided network for semantic segmentation," *IEEE Transactions on Image Processing*, vol. 30, pp. 1169–1179, 2020.

[197] Z. Huang, X. Wang, L. Huang, C. Huang, Y. Wei, and W. Liu, "Ccnet: Criss-cross attention for semantic segmentation," 2019.

[198] J. He, Z. Deng, L. Zhou, Y. Wang, and Y. Qiao, "Adaptive pyramid context network for semantic segmentation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[199] Z. Zhu, M. Xu, S. Bai, T. Huang, and X. Bai, "Asymmetric non-local neural networks for semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 593–602.

[200] J. He, Z. Deng, and Y. Qiao, "Dynamic multi-scale filters for semantic segmentation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

[201] K. E. Kim, "Adaptive majority filtering for contextual classification of remote sensing data," *International Journal of Remote Sensing*, vol. 17, no. 5, pp. 1083–1087, 1996.

[202] C. Zhan, "A hybrid line thinning approach," in *AUTOCARTO-CONFERENCE*, American Society for Photogrammetry and Remote Sensing, 1993, pp. 396–396.

[203] P. K. Saha, G. Borgefors, and G. Sanniti di Baja, "A survey on skeletonization algorithms and their applications," *Pattern Recognition Letters*, vol. 76, pp. 3–12, 2016, Special Issue on Skeletonization and its Application, ISSN: 0167-8655. DOI: https://doi.org/10.1016/j.patrec.2015.04.006. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167865515001233.

[204] F. L. Gaol, "Bresenham algorithm: Implementation and analysis in raster shape.," *Journal of Computers*, vol. 8, no. 1, pp. 69–78, 2013.

[205] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973. DOI: 10.3138/FM57-6770-U75U-7727.

[206] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 39, no. 4, pp. 640–651, 2017.