UNIVERSITÀ
DEGLI STUDI
DI PADOVA

DOCTORAL THESIS

Learning-Based Perception Methods for Cross-Domain Autonomous Robotics Navigation and Environment Monitoring

Author:
Daniel Fusaro

Supervisor:
Prof. Alberto Pretto
Co-Supervisor:
Prof. Emanuele Menegatti
Reviewers:
Dr. Nived Chebrolu
Prof. Arturo Gil Aparicio

A thesis submitted in fulfillment of the requirements
for the degree of Doctor of Philosophy in Information Engineering

at the

Department of Information Engineering (DEI)

December 1, 2025

Abstract

AUTONOMOUS mobile robots rely on robust perception to safely navigate and interact with complex, dynamic environments. Accurate understanding of the surroundings is critical not only for obstacle avoidance and traversability assessment but also for higher-level tasks such as semantic reasoning, object recognition, and temporal monitoring. Traditional approaches to robotic perception often struggle to balance accuracy, computational efficiency, and generalization across diverse scenarios, particularly in unstructured or crowded environments where terrain, vegetation, and dynamic agents introduce significant variability. Furthermore, the development of learning-based perception methods is frequently constrained by the availability of large-scale, high-quality datasets, limiting their applicability in real-world settings.

Efficient and adaptable perception solutions are therefore fundamental to enable robots to operate safely and autonomously, from urban streets and industrial settings to natural outdoor environments and indoor spaces. The key to addressing these challenges are the integration of multi-modal sensors, architectures capable of capturing fine-grained geometric and semantic information, and parameter-efficient training strategies. High-performance methods must also be deployable on embedded platforms, where computational and energy resources are limited, while remaining robust to domain shifts and variations in sensor data.

In this thesis, we address these challenges by developing novel approaches for traversability analysis, 3D semantic and instance segmentation, place recognition, and temporal object re-identification, alongside the creation of a multi-sensor, large-scale dataset for complex outdoor environments. Our contributions improve both the accuracy and efficiency of robotic perception systems, providing practical methods and resources to advance real-time, robust scene understanding in autonomous mobile robotics.

Most of the contributions presented in this thesis are available as open-source software packages. We believe that this will foster collaboration, enable precise repeatability of our experiments, and facilitate future research on robotic perception.

Acknowledgements

First of all, initially I didn't want to pursue a doctoral journey. I wanted to work in a good company or start my own business. However, during my master thesis, my (now) supervisor Prof. Alberto Pretto believed in me and gave me the opportunity to experience research for the first time. I accepted the challenge, and soon I became fascinated by it. I quickly acknowledged that Research allowed me to have fun while working on my passion, to improve myself day after day, to constantly live on the edge of and improve human knowledge. For this, and much more, For this, and much more, I am profoundly grateful for his trust, for all his suggestions, and for the continuous support he provided throughout these more than four years of collaboration.

I'm also grateful to my co-supervisor, Prof. Emanuele Menegatti, who supported me along the way and offered me many opportunities to pursue my newly discovered passion: educational robotics.

During my Ph.D. journey, I had the pleasure of collaborating extensively with Prof. Loris Nanni, who constantly encouraged me to improve myself by keeping up with the latest research. He involved me in several projects, sometimes different from my cup of tea, which greatly enriched my experience.

I also had the privilege to be a visiting Ph.D. student at the University of Bonn, in the Photogrammetry & Robotics Lab, under the supervision of Prof. Cyrill Stachniss, Dr. Jens Behley, and Dr. Tiziano Guadagnino. I learnt so much during my time there, and I am very thankful to all the members of the lab who made me feel part of the team from day one.

I want to thank all my lab mates who, day after day, made the Ph.D. journey lighter and funnier. All the FRoG guys: Simone, for all the briefings, fun challenges, and the many Golias sweets; Wanmeng "Robin", for our nice chats—often in the canteen, but really everywhere ; Emilio, Matteo, Daniele and Marco, for your great experience that you shared with me.

Matteo, for your many suggestions and your technical support; Davide, for making my days more *allegro*; Alberto, Mattia and Laura, my desk mates, who (together with many others) had to endure countless (to me) funny jokes; Laura, for sparking my interest in learning and teaching educational robotics; Edoardo, for all the great technical and non-technical conversations; Francesco Pasti, truly the best tutor I could have had as a lecturer; Tommaso and Iustin, my moka coffee mates. And obviously all the IASLab team: Alberto Gottardi, Stefano Tortora, Leonardo, Alessio, Gabriele, Anna and Piero, Paolo, Miro, Francesca, Gloria, Ashkan, and all the professors.

I want to thank two international students, with which I crossed my adventure: Gustavo, my Brazilian deep learning mate, and Antonio, for sharing the pain of dataset annotation.

And finally, I must thank my parents, Vania and Giuseppe, which supported me along the way, and my girlfriend, Carlotta, who constantly encouraged me to become a better version of myself every day, and who reminded me that love is the most important thing in life.

Contents

Glossary

AUV Autonomous Underwater Vehicle.

CAD Computer-Aided Design.

CNN Convolutional Neural Networks.

CPU Central Processing Unit.

CUDA Compute Unified Device Architecture.

FOV Field Of View.

GCNN Graph Convolutional Neural Network.

GPS Global Positioning System.

GPU Graphics Processing Unit.

IMU Inertial Measurement Unit.

KDTree K-Dimensional Tree.

LiDAR Light Detection And Ranging.

LoRA Low Rank Adaptation.

MLP Multi-layer Perceptron.

PCA Principal Component Analysis.

PEFT Parameter-Efficient Fine-Tuning.

RAM Random Access Memory.

SAM Segment Anything Model.

SEEM Segment Everything Everywhere (all-at-once) Model.

SLAM Simultaneous Localization And Mapping.

SVM Support Vector Machine.

USBL Ultra-Short BaseLine.

ViT Vision Transformer.

List of Figures

List of Tables

Chapter 1

Introduction

OVER the past two decades, mobile robotics has undergone a deep transformation, driven largely by advances in sensing, computation, and machine learning. Autonomous systems are increasingly deployed in unstructured, dynamic, and large-scale settings such as urban roads, agricultural fields, aerial and underwater domains. Robotic perception is the first task in many mobile robot applications, and is the ability of acquiring knowledge from sensor data, or in other words, to transform raw sensor measurements into scene understanding.

The field of robotic perception has particularly benefited from breakthroughs in deep learning and the growing availability of multi-modal sensors such as LiDAR, cameras, and sonar. Semantic segmentation, place recognition, and traversability analysis are now standard components of the perception pipeline. They enable robots not only to detect objects but also to interpret their environment at a higher semantic level. Nevertheless, these advances come with critical challenges. Many state-of-the-art methods require extensive labeled datasets and computational resources, making them complicated to be applied in resource-constrained platforms. Moreover, perception systems must adapt across sensor modalities and environments: the robustness needed for a self-driving car navigating an urban street is different from that required for a robot exploring a murky underwater site or monitoring a vineyard over several growing seasons.

Autonomous navigation is a core component of mobile robotics, and concerns the ability of a robot to move safely and purposefully within its environment. The most commonly used mobile robots are depicted in Figure 1.1. Unmanned ground vehicles (UGVs) are widely employed in logistics, defense, and agricultural applications, where they must navigate complex terrains and often interact with human operators or other dynamic agents. Unmanned aerial vehicles (UAVs) are increasingly used in inspection, mapping, and delivery tasks, benefiting from their ability to cover large areas efficiently and operate in environments inaccessible to ground robots. In the marine domain, autonomous underwater vehicles (AUVs) are crucial for exploration and pipeline inspection, where communication constraints and the absence of GPS make autonomy essential. Finally, autonomous electric vehicles (AEVs) provide sustainable, safe, and efficient mobility in urban environments, highlighting the convergence of robotics and intelligent transportation.

Another emerging challenge lies in efficiency and adaptability. The rapid expansion of model sizes and computational demand has driven the progress in perception benchmarks, but it has also widened the gap between research prototypes and deployable solutions. Robots deployed in the field often operate on embedded hardware, face energy limitations, and encounter out-of-distribution data, i.e. data not seen during training. Thus, there is the need for new methods that are not only accurate but also power- and data-efficient, and generalizable across sensing domains. Models trained with limited data, zero-shot learning techniques, and domain adaptation strategies are promising avenues to address these challenges.

FIGURE 1.1: At the top, from left to right, are an unmanned ground vehicle (UGV) with manipulator (*Coustesy of Robotnik*), an autonomous underwater vehicle (AUV) (FlatFish, *Coutesy of Saipem*). Below are, from left to right, an unmanned aerial vehicle (UAV) (*Coustesy of DJI*) and an autonomous electric vehicle (AEV) (*Coustesy of Waymo*), serving as a robotaxi in San Francisco.

The development and evaluation of perception methods heavily depends on high-quality datasets. Although many benchmarks are public, they often remain closely tied to specific application areas or are not multi-modal. As robotics expands into more diverse environments, there is a significant need for datasets and approaches that not only reflects these complexities but also facilitates the progress in robust and general perception.

1.1 Main contributions

The main contributions of this thesis are novel approaches to extend the current capabilities of robotic perception exploiting learning-based techniques. In particular, we focused on small-data, zero-shot, and computationally efficient methods that can operate in real time on resource-constrained platforms, and generalize across sensing modalities and environments.

We first briefly introduce the fundamental concepts used in this thesis in Chapter 2. These include: 2D and 3D vision sensors used in robotics, the definition of the main learning-based vision problems, an overview of the major deep learning-based 2D and 3D vision models, focusing on semantic, instance and panoptic segmentation, describing traversability analysis for mobile robot navigation as a special case of semantic segmentation, and finally sonar-based underwater visual place recognition methods. These are the foundation of our approaches.

In Chapter 3, we deal with efficient traversability analysis. It's a critical task in which a robot must be able to distinguish safe (traversable) from unsafe (not traversable) regions quickly and accurately to perform navigation and planning. Many existing methods rely heavily on GPU acceleration, which is not always available in embedded or field-deployed platforms. Thus, we propose P-SVM, a pyramidal 3D feature fusion framework on polar grids for CPU-based traversability analysis, addressing the need for robust yet computationally efficient navigation strategies. Our method is data- and power- efficient, consuming less than 70% electring power with respect to competitive deep learning approaches, and real-time

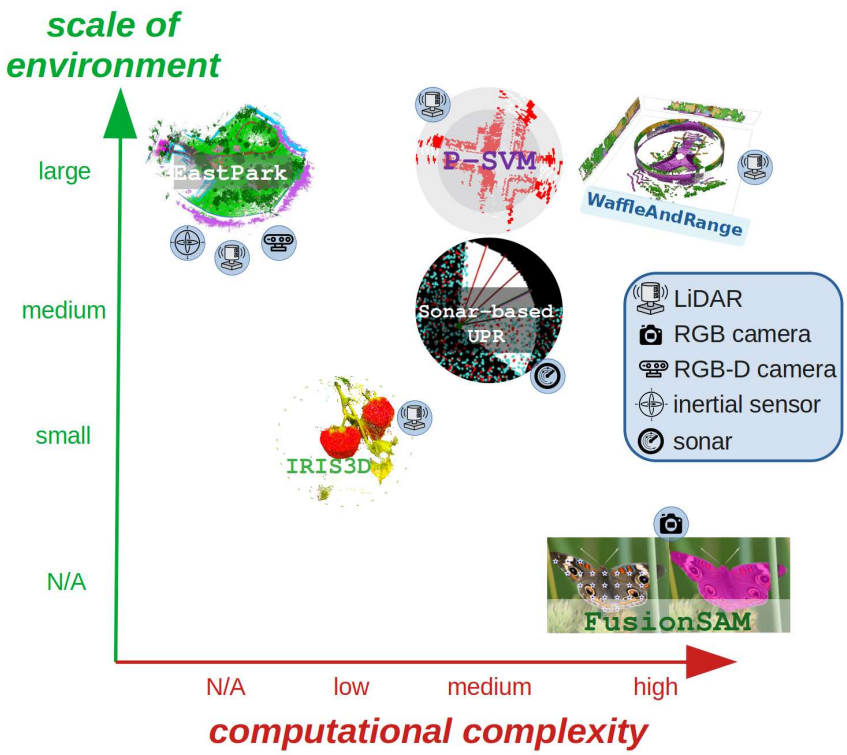


FIGURE 1.2: An overview of the main contributions presented in the thesis, describing the computational complexity along the x-axis, the scale of environment of application along the y-axis, and the sensor modality (indicated by the icons, according to the legend on the right).

both in desktop CPUs and embedded devices. This makes our method suitable for practical, real-world applications where computational resources are limited. P-SVM can be applied it to different real robotic platforms, including UGVs and UAVs. As shown in Figure 1.2, P-SVM operates in large-scale environments with medium computational complexity, thanks to the reliance on support vector machines processing LiDAR point clouds.

Building on the theme of efficiency, in Chapter 4 we tackle the next challenge, which arises in semantic understanding of 3D LiDAR point clouds. Although semantic segmentation has shown great potential, most methods require large labeled datasets and GPU resources. In field robotics, however, data can be scarce, and real-time constraints are pressing. To bridge this gap, we propose WaffleAndRange, a method that exploits local features and range image representations to achieve real-time point cloud semantic segmentation with limited training data, i.e. small data. Small data means that we can train our model with only a few annotated samples, making it feasible to deploy in scenarios where collecting large datasets is impractical. This enables LiDAR semantic segmentation also in scenarios where both computational and data resources are restricted, such as particular areas not commonly covered by public datasets. In Figure 1.2, WaffleAndRange is placed among methods operating in large-scale environments, with high computational complexity due to the deep learning-based model processing LiDAR point clouds.

Complementary to LiDAR, vision-based perception remains an essential tool in robotics. However, the performance of traditional 2D image semantic segmentation networks can degrade when transferred across domains. Recently, large-scale zero-shot segmentation models have emerged, offering strong generalization without task-specific training. Zero-shot training refers to the ability of a model to correctly make predictions on classes that were not seen during training. In Chapter 5 we explore how zero-shot segmentators can be exploited to improve existing 2D segmentators, addressing the need for generalizing vision systems that

remain reliable in unseen environments without expensive re-annotation or re-training. In multiple datasets we demonstrate that our approach, SAMFusion, can consistently enhance the performance of existing segmentators, especially in cross-domain scenarios, obtaining state-of-the-art results. In Figure 1.2, SAMFusion is a method with high computational complexity, due to the usage of large vision-based deep learning models (e.g. SAM, SEEM) tailored for RGB image data.

Progress in robotic perception depends heavily on high-quality datasets. Existing datasets are often limited to specific domains or lack the scale and multi-modality required to benchmark modern algorithms. To this end, in Chapter 6 we introduce EastPark, a multi-sensor, large-scale dataset designed for 3D LiDAR semantic segmentation, along with a study of parameter-efficient fine-tuning methods. This work directly addresses the need for a dataset comprising a variety of ground surfaces (e.g., grass, gravel, mud), dynamical agents such as people and vehicles and irregular natural structures such as vegetation, bushes and trees. We collected this dataset in a park near Padua with a UGV equipped with a 64-channel LiDAR, an RGB-D camera, and an IMU, as shown in Figure 1.2. With this dataset, we aim to facilitate the development and evaluation of robust 3D perception methods that can generalize across diverse outdoor environments.

Indeed, robotic perception also plays a fundamental role in agriculture, where the themes of data- and power-efficiency and high-quality data align with the previous works. In Chapter 7 we address fruit growth monitoring over time that requires not only accurate 3D perception but also temporal re-identification of individual fruits. We present IRIS3D, an approach for temporal horticultural fruit monitoring via 3D instance segmentation and re-identification using colored point clouds, directly addressing the need in agricultural robotics for automated, fine-grained, and long-term crop monitoring. IRIS3D features low computational complexity thanks to the reduced size of the deep learning models and operates in small-scale environments (e.g. orchards, fields), as shown in Figure 1.2. Colored point clouds are obtained by a terrestrial laser scanner (TLS), which is a particular type of LiDAR sensor. With this work, we aim to provide farmers with detailed insights into fruit development, enabling better management practices and yield optimization.

Finally, robotics perception is not confined to terrestrial domains. In Chapter 8 we address robotics when operating conditions are even more extreme. In underwater environments, visibility conditions, such as water turbidity and scarce light, often preclude optical sensors, and sonar becomes the primary sensing modality. Reliable place recognition is essential for navigation and mapping in such conditions. Yet, sonar data is inherently noisy and structurally different from vision or LiDAR. We developed a learning-based framework for real-time underwater place recognition using multibeam sonar, evaluated in both synthetic and real-world environments. This work was motivated by the demand for robust localization in challenging, GPS-denied underwater environment. As in Figure 1.2, our sonar-based underwater place recognition method operates in medium-sized environments and has a medium computational complexity.

Each chapter starts with an Introduction section that outlines the chapter's content in relation to the main topic of the thesis. Then, we provide a Motivation section, which highlights the specific problem under consideration and its relevance to the field of robotics. A literature review then positions the work with respect to existing research, after which the chapter presents its scientific contribution. Finally, the proposed methods are subsequently assessed in an experimental evaluation, and the chapter concludes with a summary that synthesizes the key findings.

1.2 Publications

Part of this thesis has been published in peer-reviewed conference and journal articles:

- **D. Fusaro**, E. Olivastri, I. Donadi, D. Evangelista, E. Menegatti, and A. Pretto. Pyramidal 3d feature fusion on polar grids for fast and robust traversability analysis on cpu. Robotics and Autonomous Systems (RAS), vol. 170, p. 104 524, 2023
- **D. Fusaro**, S. Mosco, E. Menegatti, and A. Pretto. Exploiting local features and range images for small data real-time point cloud semantic segmentation. IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2024, pp. 4980–4987.
- L. Nanni, **D. Fusaro**, C. Fantozzi, and A. Pretto, "Improving existing segmentators performance with zero-shot segmentators," Entropy, vol. 25, no. 11, 2023.
- **D. Fusaro**, F. Magistri, J. Behley, A. Pretto, and C. Stachniss, "Horticultural Temporal Fruit Monitoring via 3D Instance Segmentation and Re-Identification using Colored Point Clouds," Computers and Electronics in Agriculture (COMPAG), 2025 (submitted)
- **D. Fusaro**, S. Mosco, W. Li, and A. Pretto, "Real-time underwater place recognition in synthetic and real environments using multibeam sonar and learning-based descriptors," in IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR), 2025, pp. 1–6.

The following are publications I was involved in during my doctorate as a collaborator but they are not part of this thesis:

- L. Lobefaro, M. Sodano, **D. Fusaro**, F. Magistri, M. V. Malladi, T. Guadagnino, and C. Stachniss. Spatio-Temporal Consistent Semantic Mapping for Robotics Fruit Growth Monitoring. IEEE Robotics and Automation Letters (RAL), 2025.
- W. Li, **D. Fusaro**, E. Olivastri, S. Mosco, N. Bellotto, and A. Pretto. P-SVM2: Enhancing lidar-based traversability analysis with augmented point cloud descriptor for autonomous mobile systems. IEEE International Conference on Cybernetics and Intelligent Systems (CIS) and IEEE International Conference on Robotics, Automation and Mechatronics (RAM), 2024, pp. 352–359
- S. Mosco, **D. Fusaro**, W. Li, E. Menegatti, and A. Pretto. Point-Plane Projections for Accurate LiDAR Semantic Segmentation in Small Data Scenarios. Computer Vision and Image Understanding (CVIU), 2025 (under review)
- S. Mosco, W. Li, **D. Fusaro**, and A. Pretto. Revisiting Retentive Networks for Fast Range-View 3D LiDAR Semantic Segmentation,. IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), 2026 (accepted).
- W. Li, S. Mosco, **D. Fusaro**, and A. Pretto. Confidence-Guided Pseudo-Label Sampling for Unsupervised Domain Adaptation in 3D LiDAR Semantic Segmentation. European Conference on Mobile Robots (ECMR), 2025 (accepted).
- L. Nanni, M. D. Gobbi, R. D. A. M. Junior, and **D. Fusaro**. Advancing taxonomy with machine learning: A hybrid ensemble for species and genus classification. Algorithms, vol. 18, no. 2, p. 105, 2025.
- L. Nanni, N. Maritan, **D. Fusaro**, S. Brahnam, F. B. Meneguolo, and M. Sgaravatto. Insect identification by combining different neural networks. Expert Systems with Applications, vol. 273, p. 126 935, 2025.

1.3 Open-Source Contributions

Some of the algorithms and techniques presented in this thesis are also supported by open-source software implementation to facilitate future research:

- Chapter 3 presents our proposed pyramidal 3D feature fusion framework on polar grids for CPU-based traversability analysis. The implementation is available at https://github.com/Bender97/TraversabilityAnalysis.
- Chapter 4 presents our proposed method that exploits local features and range image representations to achieve real-time point cloud semantic segmentation with limited training data. The implementation is available at https://github.com/Bender97/LocalFeaturesSemSeg.
- Chapter 5 presents our proposed method that expoits zero-shot image segmentators to improve existing 2D image segmentators. The implementation is available at https://github.com/LorisNanni/Improving-existing-segmentators-performance-with-zero-shot-segmentators.
- Chapter 6 presents EastPark, a multi-sensor, large-scale dataset that we designed for 3D LiDAR semantic segmentation, along with a study of parameter-efficient fine-tuning methods. Our dataset, trained weights and code are available at https://zenodo.org/records/17121785.
- Chapter 7 presents our proposed method for temporal horticultural fruit monitoring via 3D instance segmentation and re-identification using colored point clouds. The implementation is available at https://github.com/PRBonn/IRIS3D.

Chapter 2

Fundamentals

THE main topic of this thesis, robotic perception, builds upon well-known concepts in robotics and scene understanding, with particular emphasis on the machine learning field. This chapter is organized as follows. We begin by introducing the main 2D and 3D vision sensors used in robotics. We then describe the main learning-based vision problems, including classification, object detection, segmentation and place recognition. Next, we describe deep learning-based 2D and 3D Vision Models, focusing on semantic segmentation. Following we chaaracterize the main methods for 3D point cloud instance and panoptic segmentation methods, both for indoor and outdoor applications. We introduce traversability analysis for mobile robot navigation as a special case of semantic segmentation. Finally, we describe sonar-based underwater visual place recognition methods.

2.1 2D and 3D Vision Sensors

This section provides an overview of the primary 2D and 3D vision sensors utilized in robotics. These sensors enable robots to perceive and interpret their surroundings, allowing tasks such as navigation, object detection, and scene understanding. We will discuss the operating principles, advantages, applications and limitations of various sensor types, including digital cameras, RGB-D cameras, sonars, and LiDARs.

2.1.1 Digital Cameras

Digital cameras are devices able to produce two-dimensional (2D) arrays (called images) of measurements of a specific electromagnetic radiation (e.g., the visible light) coming from non-occluded surfaces of a framed three-dimensional (3D) scene. In this 3D-2D projection, one dimension (i.e., the points' depth) is anyhow lost. The basic components of a digital camera are (a) an imaging sensor; (b) an optical system, called lens, used to route the sensed information (e.g., the visible light) from each 3D point toward a 2D point of the imaging sensor (see Figure 2.1, right); (c) an internal Analog to Digital Converter (ADC) used to convert each pixel measurement into a digital value.

In nature, each 3D point emits radiations that spread in all directions in space, potentially reaching the entire area of an imaging sensor exposed to such radiations without a lens (see Figure 2.2, left). Ideally, for each 3D point emitting omnidirectional radiations, there is a single 2D point on the imaging sensor that captures exactly one ray of this radiation and no others. This ideal model is known as the pinhole model.

The purpose of the lens is to closely approximate the ideal scenario by directing multiple radiation rays originating from the same 3D point toward a single 2D point on the imaging sensor (see Figure 2.2, right). This is further achieved by incorporating a barrier with a central aperture (known as the iris) to block the majority of the rays. While this model (known as thin lens model) provides only an approximation of the ideal scenario (e.g., the lower 3D point

[t]

FIGURE 2.1: Two examples of digital, area scan cameras. A Basler Dart without lens, with visible CMOS imaging sensor (left): an IDS uEye with mounted lens. *(Courtesy of Basler and IDS Imaging.)*

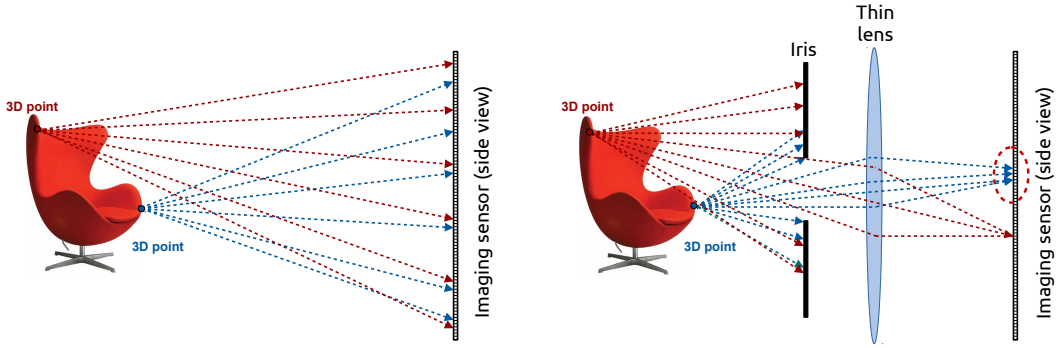


FIGURE 2.2: Without a lens, each 3D point of the scene will project its radiation into each point of the 2D imaging sensor, producing a useless image (left); A lens in front of the sensor (right), here represented by its simplest model, the thin lens model, is used to mitigate as possible this phenomenon.

on the right side of Figure 2.2 is projected onto a small region on the 2D imaging sensor), this model, along with the pinhole model, remains widely used for describing and modeling real cameras. Cameras that are described by the pinhole model are also called perspective cameras.

Digital cameras can be used in a very wide range of applications in mobile robotics, among others: place recognition, visual servoing, ego-motion estimation, SLAM (Simultaneous Localization and Mapping), object detection and classification, etc.

2.1.2 RGB-D Cameras

RGB-D (RGB-Depth) cameras provide both color images and depth estimates. They have become increasingly popular in robotics due to their ability to capture rich 3D information about the environment in a compact and affordable form factor. Examples of RGB-D cameras are shown in Figure 2.3.

The depth information is typically obtained using one of three main technologies:

- **Time-of-Flight (ToF):** These cameras emit modulated infrared light and measure the phase shift or time delay of the reflected light to compute depth. ToF cameras can provide real-time depth maps with good accuracy and are effective in various lighting conditions.

FIGURE 2.3: Three example of RGB-D cameras: the Intel® RealSense D435 (left); the Microsoft® Kinect II (center); the Microsoft® Azure Kinect (right). Both the Microsoft® Kinect II and Azure Kinect use a CW ToF camera as ranging sensor, while the D435 uses active stereo vision *(courtesy of Intel® and Microsoft®).*

- **Passive Stereo Vision:** This methods use two or more cameras (usually identical) to capture images from slightly different viewpoints. By analyzing the disparities between corresponding points in the images, depth can be inferred through triangulation. Passive stereo systems rely on ambient light and texture in the scene, making them less effective in low-light or textureless environments.
- **Active Stereo Vision:** Active stereo systems use two or more cameras along with an infrared projector that emits a random dot pattern onto the scene. The cameras capture images of the scene, and depth is computed by finding correspondences between the images. This method combines the benefits of stereo vision with active illumination to improve performance in low-texture areas.
- **Structured Light (SL):** SL cameras project a known pattern of light onto the scene and use a single camera to capture the deformation of this pattern. By analyzing the distortion, the system can infer depth information. SL cameras are often used in indoor environments and can achieve high resolution.

Passive stereo cameras are natively RGB-D sensors, while other RGB-D sensors are ensembles composed of a ToF, a SL, or an active stereo camera rigidly coupled in the same chassis with a color camera (see Figure 2.3).

The transformation that relates the two sensors is known, so it is possible to acquire depth maps that are registered with the related RGB images acquired by the color camera. The registration is the process of associating each pixel of the RGB image with a corresponding pixel in the depth map: both pixels represent projections of the same 3D point into both sensors. Then, it's possible to generate a colored point cloud that encodes both the structure and the visual appearance (i.e., the color) of the framed scene. This multimodal information enables one to tackle in a more effective way complex perception tasks such as recognizing and locating objects, 3D reconstruction of environments, and detection and tracking of people. Recent RGB-D sensors as the Microsoft Azure Kinect (see Figure 2.3, right) also integrates microphone arrays for speech and sound capture and IMUs for sensor orientation tracking.

2.1.3 Sonar

Sonar (Sound Navigation and Ranging) are sensors that use sound waves to detect objects and measure distances. Passive sonar devices only perceive sounds, while active sonars also emit pulses of sound and are commonly used in mobile robots for obstacle avoidance and self-localization. Active sonars consist of an emitter/receiver pair measuring the distance to an obstacle from the round-trip time of an emitted pulse (Figure 2.4, left). The cone of the emitted signal (sound, usually ultrasound) can be varied: the narrower the cone, the higher the angular resolution of the sensor. In order to have multiple readings at different directions,

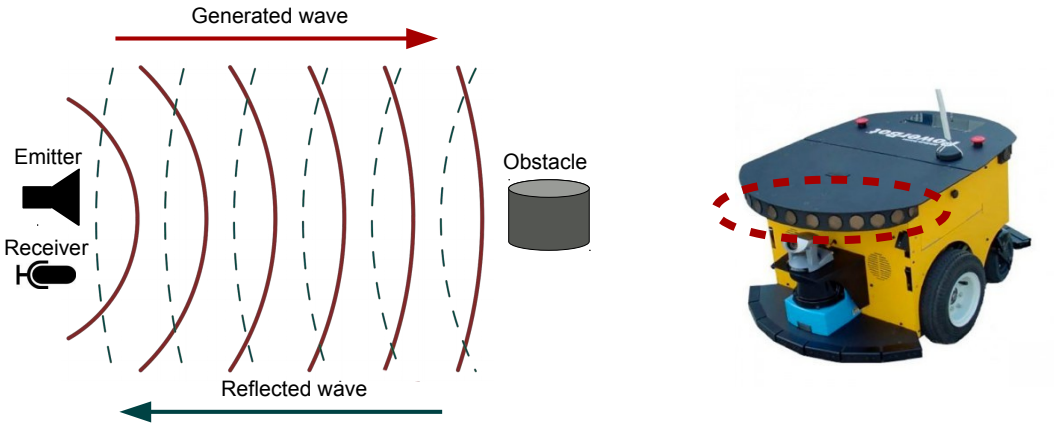


FIGURE 2.4: Operating principle of sonars (left); The red dashed ellipse highlights the array of sonars that equip the MobileRobots PowerBot platform (right) *(Courtesy of Omron Adept MobileRobots PowerBot™).*

multiple sonar devices need to be configured in an array (e.g., Figure 2.4, right).

Multibeam sonars emit multiple acoustic beams simultaneously, covering a wide angular range and providing detailed information about the underwater environment. This makes them indispensable for tasks such as bathymetric mapping, obstacle detection, and navigation in underwater robotics. The operating principle of multibeam sonar is based on the emission of acoustic pulses and the measurement of the time it takes for the echoes to return from objects in the environment. The key components of a multibeam sonar system include:

- **Transducer Array:** A set of transducers arranged in a linear or planar configuration to emit and receive acoustic signals.
- **Beamforming:** A signal processing technique used to steer and shape the acoustic beams, improving spatial resolution and enabling the simultaneous measurement of multiple directions.
- **Time-of-Flight Measurement:** The distance to an object is calculated based on the time delay between the emission of the acoustic pulse and the reception of the echo, using the speed of sound in water.

Multibeam sonar systems can generate high-resolution 2D or 3D maps of the underwater environment by combining the distance measurements from multiple beams. The resulting data can be represented as a point cloud or a grid map, which can be used for various applications in underwater robotics.

2.1.4 Light Detection and Ranging (LiDAR)

LiDAR (Light Detection and Ranging) has become a cornerstone technology in robotic perception, providing accurate and dense measurements of the surrounding environment. Unlike cameras, which capture color and texture, LiDAR sensors deliver precise geometric information in the form of point clouds. They differ from RGB-D cameras by offering longer range, higher accuracy, and robustness to lighting conditions. This enables robots to reason about structure, scale, and distance, even in conditions where visual cues are unreliable, such as poor lighting or adverse weather, where cameras fail. LiDAR perception has therefore been widely adopted in applications ranging from autonomous driving to field robotics and mapping. Examples of various LiDAR sensors are shown in Figure 2.5.

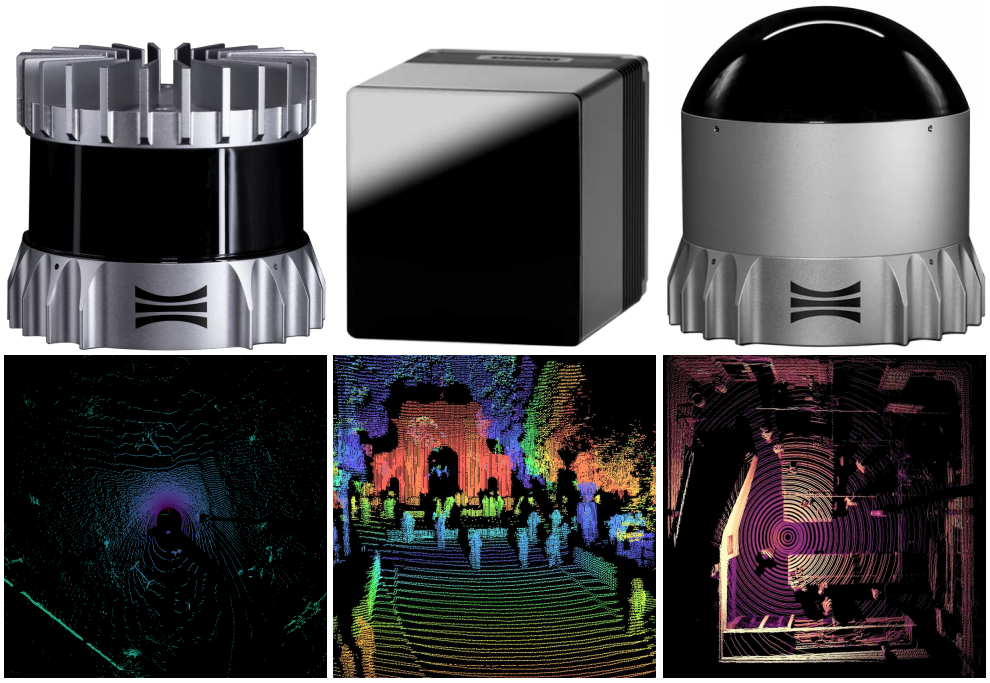


FIGURE 2.5: At the top, from left to right, are a spinning 3D LiDAR Ouster OS-1 with 64 channels, a solid-state 3D LiDAR Hesai 120FT and a solid-state 3D LiDAR Ouster OSDome with 128-channel. Below are the corresponding 3D point cloud examples produced by each sensor. colors indicate radial distance from the sensor. *Coustesy of Ouster and Hesai*.

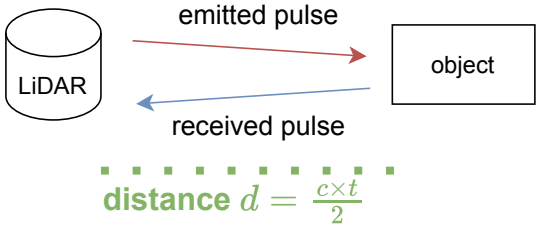


FIGURE 2.6: Illustration of LiDAR sensing: a laser emits pulses that reflect off surfaces, measuring distance by calculating the time-of-flight to generate a point cloud of the environment.

LiDAR sensors measure distances by emitting laser pulses and recording the time it takes for the reflected light to return. Each measurement corresponds to a 3D point in space, and the collection of all such points forms a 3D point cloud. Depending on the sensor design, point clouds may vary in density, field of view, and resolution.

Spinning LiDARs, such as those commonly used in autonomous driving, employ rotating laser beams to capture 360-degree scans, offering wide coverage but with non-uniform point distribution. These sensors estimate the distance to surrounding surfaces by measuring the time-of-flight (ToF) of laser pulses. Let a laser beam be emitted at time t_0 and reflected back at time t_r. The round-trip time is $\Delta t = t_r - t_0$, and the distance to the target point is:

$$d = \frac{c\,\Delta t}{2}, \tag{2.1}$$

where c is the speed of light in air. Each measurement is associated with an azimuth angle θ, determined by the rotation of the sensor, and an elevation angle ϕ, defined by the vertical arrangement of the laser channels. A point in Cartesian coordinates (x, y, z) is obtained

FIGURE 2.7: Two examples of terrestrial laser scanners (TLS): a FARO® scanner (left) and a Topcon® GTL-1200 (right) *(Courtesy of FARO® and Topcon®).*

through spherical-to-Cartesian transformation:

$$\begin{aligned} x &= d\cos\phi\cos\theta, \\ y &= d\cos\phi\sin\theta, \\ z &= d\sin\phi. \end{aligned} \tag{2.2}$$

Over one full rotation of the sensor, multiple vertical channels indexed by $i \in \{1,\ldots,N\}$ produce measurements at fixed elevations ϕ_i, generating a dense 3D point cloud:

$$\mathcal{P} = \{(x_{i,k}, y_{i,k}, z_{i,k}) \mid i \in [1,N],\, k \in [1,K]\}, \tag{2.3}$$

where k denotes the azimuth sample index. The angular resolution $\Delta\theta$ depends on the motor speed of the spinning LiDAR and the sampling frequency of the receiver, while the vertical resolution is determined by the number and spacing of the laser channels. Together, these parameters define the density and uniformity of the resulting point cloud. See Figure 2.6 for a visualization of the LiDAR sensing principle.

Solid-state LiDARs eliminate mechanical components by relying on technologies such as microelectromechanical systems (MEMS) mirrors, optical phased arrays, or flash illumination, resulting in more compact, durable, and potentially lower-cost devices, though often with narrower fields of view. Among the solid-state LiDAR, an emerging class is the dome-based LiDARs, such as the OSDome, which are designed to provide hemispherical coverage in a compact form factor, enabling dense, wide-angle perception suitable for navigation in cluttered environments, thus removing the narrower field of view limitation. As can be seen in Figure 2.5, dome-based LiDARs are often used to produce bird-eye-views, thus the typical field of application is for indoor surveillance and unmanned aerial vehicles (UAV, e.g., drones and quadcopters).

Terrestrial Laser Scanners (TLS) are high-precision LiDAR systems designed for detailed 3D mapping and surveying applications.

Unlike spinning or solid-state LiDARs, which are often mounted on mobile platforms for real-time perception, TLS units are typically stationary and used for static scans of environments (see the two examples of TLSs in Figure 2.7). However, they're generally more expensive but also more precise and capable of capturing finer details over larger distances. TLS systems often employ high-resolution laser beams and advanced scanning mechanisms to achieve dense point clouds with millimeter-level accuracy, often providing also the color information of the scanned surfaces by integrating RGB cameras. Applications of this sensor include topographic mapping, architectural documentation, and industrial inspection, as well as surveying, forestry, archaeology and cultural heritage preservation.

2.2 Learning-based Vision Problems

Perception is a fundamental task in autonomous robotics, enabling robots to interpret sensory inputs and reason about the structure of their environment. Modern perception systems are dominated by learning-based methods, primarily deep neural networks trained on large annotated datasets. These methods rely on general machine learning concepts such as optimization, loss functions, and neural network architectures, which are assumed to be known to the reader. Traditional computer vision techniques once relied on hand-crafted features and shallow classifiers, but the advent of deep learning has revolutionized the field, establishing convolutional neural networks (CNNs) as the dominant paradigm.

In this section, we focus on the specific problem formulations in computer vision that are most relevant to robotic perception, describing their mathematical structure, objectives, and evaluation criteria. We begin with classification as the canonical supervised learning problem in vision. Building upon it, we cover object detection and the different levels of segmentation: semantic, instance, and panoptic segmentation. Finally, we discuss the learning paradigms underlying these tasks: standard supervised training as well as one-shot, few-shot, and zero-shot learning, which have become increasingly important for data-scarce domains.

2.2.1 Classification

In classification, the goal is to assign a single semantic label to an input signal, that for vision problems can be an image or a point cloud. In the most common scenario, the labels are taken to be disjoint, so that each input signal is assigned to one and only one label [10].

We will describe the classification problem in the context of image classification, but the same principles apply to other modalities such as point clouds.

Let an image be denoted as $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$, with height H, width W, and C channels. The ground-truth label is a discrete variable $y \in Y = \{1, \ldots, L\}$, where L is the number of classes. When $L = 2$ it's a binary classification problem, when $L > 2$ it's a multiclass, or multilabel, classification problem.

A classifier is a discriminant function that assigns each image $\mathbf{I}$ to a specific class. Traditional discriminative function classifiers are,, for instance, support vector machine (SVMs). Instead of tackling the problem directly, a more powerful classwhich produces a probability distribution over the classes $\hat{\mathbf{y}} = [\hat{y_1}, \ldots, \hat{y_L}]$ when composed withification-approach models the conditional probability distributions $\hat{p}(\hat{y}_c \mid \mathbf{I})$ in an inference stage and subsequently uses this distributions to make optimal decisions. In this setting, a classifier is a discriminative probabilistic model $\mathbf{f}_\theta : \mathbf{I} \rightarrow \hat{p}(\hat{\mathbf{y}})$. It is obtained by composing a feature extraction parametric model, $\mathbf{f}_{e,\theta} : \mathbf{I} \rightarrow \hat{\mathbf{y}}_{\mathbf{log}}$ parameterized by θ, and the softmax function:

$$\hat{p}(\hat{\mathbf{y}} \mid \mathbf{I}) = \text{softmax}(\mathbf{f}_{e,\theta}(\mathbf{I})). \tag{2.4}$$

The vector $\hat{\mathbf{y}}_{\mathbf{log}}$ is called logits of the input $\mathbf{I}$, and is often used to fuse different classifiers output into a single output, creating a so-called ensemble. Ensemble classifiers tend to perform better than single classifiers. Logits are often fused using the weighted rule, which assigns different weights to different classifiers.

The predicted class $\hat{y}$ is obtained as:

$$\hat{y} = \text{argmax}\, \hat{p}(\hat{\mathbf{y}} \mid \mathbf{I}). \tag{2.5}$$

The parametric model can be optimized (i.e., trained) using a training set (split into actual training and validation) and evaluated usign a test set, containing examples that have never been seen by the model during training. The training process aims to minimize an error function, also called loss function. The standard loss to train classifiers is the categorical

$$TP_c = |\{i \mid \hat{y}_i = c, y_i = c\}| \quad FP_c = |\{i \mid \hat{y}_i = c, y_i \neq c\}|$$
$$FN_c = |\{i \mid \hat{y}_i \neq c, y_i = c\}| \quad TN_c = |\{i \mid \hat{y}_i \neq c, y_i \neq c\}|.$$

$$\mathrm{Pr}_c = \frac{TP}{TP+FP} \quad \mathrm{Re}_c = \frac{TP}{TP+FN}.$$

cross-entropy:

$$\mathcal{L}_{cls} = -\log \hat{p}(\hat{\mathbf{y}} \mid \mathbf{I}). \tag{2.6}$$

The most used metric to evaluate classifiers is accuracy:

$$\text{accuracy} = \frac{|\{i \mid \hat{y}_i = y_i\}|}{N}, \tag{2.7}$$

where N is the number of test samples and $|\cdot|$ indicates the cardinality of the set argument.

Accuracy is a good metric when dealing with balanced data, but it is unreliable for imbalanced data. For example, in a binary classification in which the negative class is predominant, a model that predicts all instances as negative can achieve high accuracy but miss all actual cases.

The most used metrics in classification handling imbalances are precision, recall, intersection over union and F1 score. They are based on the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN), and are computed for each class $c \in Y$:

Precision (Pr) is the ratio of correctly identified positive instances out of all instances predicted as positive, and recall (Re) is the ratio of correctly identified positive instances out of all actually positive instances:

Intersection over union (IoU), also known as the Jaccard index, measures the ratio between the number of correct positive predictions and the number of occurrences of the positive class. Formally, for each class $c \in Y$:

$$IoU_c = \frac{TP_c}{TP_c + FN_c + FP_c}. \tag{2.8}$$

F1 score reflects a model's ability to both correctly identify positive instances and retrieve all actual positive cases:

$$F1_c = \frac{2TP_c}{2TP_c + FN_c + FP_c} = \frac{2\mathrm{Pr} \times \mathrm{Re}}{\mathrm{Pr} + \mathrm{Re}}. \tag{2.9}$$

F1 score is the harmonic mean between precision and recall: it penalizes extreme values, meaning a high F1 score requires both precision and recall to be relatively high.

In multilabel classification problems, these metrics are usually averaged to reflect the classifier performance over all L classes. In particular, the mean intersection over union (mIoU) and mean F1 score (mF1) are the most used:

2.2.2 Object Detection

Object detection extends classification to localize multiple objects within an input signal, such as an image or a point cloud. Again, we describe the problem in the context of image-based object detection, but the same principles apply to other modalities.

The ground truth consists of a set of labeled bounding boxes:

$$\mathcal{B} = \{(\mathbf{b_j}, c_j)\}_{j=1}^{N}, \tag{2.10}$$

$$\text{mIoU} = \frac{\sum_{c=1}^{L} \text{IoU}_c}{L} \quad \text{mF1} = \frac{\sum_{c=1}^{L} \text{F1}_c}{L}.$$

where each $\mathbf{b_j} \in \mathbb{R}^4$ encodes a box (parameterized by its 2D center, width and height) and $c_j \in Y$ is its class.

A detector produces:

$$\hat{\mathcal{B}} = \{(\hat{\mathbf{b}}_\mathbf{i}, \hat{c}_i, \hat{s}_i)\}_{i=1}^{\hat{N}}, \tag{2.11}$$

where each $\hat{\mathbf{b}}_\mathbf{i} \in \mathbb{R}^4$ is a predicted box, $\hat{c}_i \in Y$ its class, and $\hat{s}_i \in [0,1]$ a confidence score.

A detector is typically trained by minimizing a weighted combination of classification and localization losses:

$$\mathcal{L} = \mathcal{L}_{cls} + \lambda \mathcal{L}_{loc}, \tag{2.12}$$

with $\mathcal{L}_{cls}$ a cross-entropy or focal loss and $\mathcal{L}_{loc}$ a regression loss (e.g., ℓ_1-loss).

The standard metric is average precision (AP), computed for each class $c \in Y$. It quantifies the precision-recall trade-off. To compute AP, we must match predicted boxes with ground truth boxes: a prediction box $(\hat{\mathbf{b}}_\mathbf{i}, \hat{c}_i)$ matches a ground truth box $(\mathbf{b_j}, c_j)$ if they share the same class and:

$$\text{IoU}(\hat{\mathbf{b}}_\mathbf{i}, \mathbf{b_j}) = \frac{|\hat{\mathbf{b}}_\mathbf{i} \cap \mathbf{b_j}|}{|\hat{\mathbf{b}}_\mathbf{i} \cup \mathbf{b_j}|} \geq \tau \tag{2.13}$$

for a threshold τ (usually 0.5). Then, the predicted box is a true positive. If the predicted box does not match any ground-truth box, it is a false positive. If a ground-truth object has no matching predicted box, it is a false negative.

We sort all predicted boxes by the confidence score in descending order, and calculate precision and recall at various confidence score thresholds. AP is the area under the precision-recall curve, and it is typically computed as the mean precision over all recall levels or through interpolation.

The mean average precision (mAP) averages the AP across classes. When $\tau = 0.5$ is used, we refer to mAP as mAP@50. Accordingly, for a $\tau = 0.95$, we use mAP@95.

2.2.3 Segmentation

In classification problem, an entire input signal is assigned to a single class label. Segmentation problems such as semantic, instance and panoptic segmentation extend classification to individual pixels within an image or points of a point cloud. See Figure 2.8 for examples of 2D image segmentation.

Semantic segmentation is the basic within the segmentation problems.

Given an image $\mathbf{I} \in \mathbb{R}^{H \times W \times C}$, a semantic probabilistic segmentor produces a probabilistic segmentation $\hat{\mathbf{S}} \in [0,1]^{H \times W \times L}$. From S, we can obtain a segmentation mask $\hat{\mathbf{M}} \in Y^{H \times W}$ (see Figure 2.8, top-right), in which each pixel is assigned the class having the highest probability along the L dimension.

Analogously, given an input point cloud $\mathcal{P} = \{(x_i, y_i, z_i)\}_{i=1}^{N}$, a semantic probabilistic segmentor produces a probabilistic segmentation $\hat{\mathbf{M}}_\mathbf{pc} \in [0,1]^{N \times L}$.

We continue the description in the context of image-based semantic segmentation, but the same principles apply to point clouds.

Particularly useful is to decompose the segmentation mask into L binary segmentation masks $\{\hat{\mathbf{M}}_\mathbf{i}\}_{i=1}^{L}$, in which all pixels are zero except for the pixels having class i.

A segmentor is typically trained by minimizing a weighted combination of categorical cross-entropy loss $\mathcal{L}_{cls}$, focal loss $\mathcal{L}_{focal}$ and Lovász-Softmax [8] loss $\mathcal{L}_{lov}$:

$$\mathcal{L} = \lambda_{cls}\mathcal{L}_{cls} + \lambda_{focal}\mathcal{L}_{focal} + \lambda_{lov}\mathcal{L}_{lov}, \tag{2.14}$$

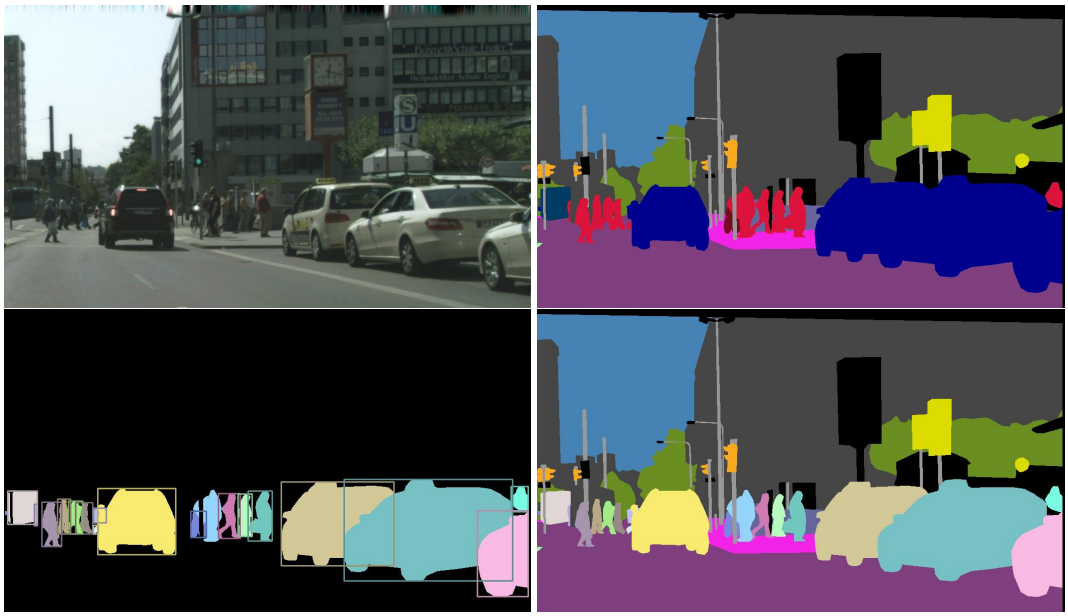


FIGURE 2.8: Examples of 2D image segmentation [85]: original image (top-left); semantic segmentation (top-right); instance segmentation (bottom-left); panoptic segmentation (bottom-right). *Courtesy of Cityscapes [29].*

.

with $\lambda_{cls}, \lambda_{focal}, \lambda_{lov}$ their weight, respectively. Semantic segmentation only cares about semantic labels, i.e. it does not differentiate different objects sharing the same class.

The standard metric for semantic segmentation is per-class IoU, and averaged across all classes mIoU. For a class $c \in Y$ and ground truth binary segmentation masks $\{\mathbf{M_i}\}_{i=1}^{L}$, the IoU is defined as:

$$\text{IoU}_c(\hat{M}_c, M_c) = \frac{|\hat{M}_c \cup M_c|}{|\hat{M}_c \cap M_c|}. \tag{2.15}$$

Instance segmentation recognizes the pixel-level regions of each object instance in an image, and point-level regions of each object instance in a point cloud. It is kwnown as simultaneous detection and segmentation, and differently from semantic segmentation, it assigns a label (class + instance ID) to each pixel of $\mathcal{I}$ or points of $\mathcal{P}$, but it considers only particular classes called "thing" (e.g. car, people, animal, truck) All the other pixels or points are assigned the background class, a semantic label indicating no object. The background class is usually ignored in loss and performance analysis.

In the context of images (the point cloud case is analogous), the ground truth is a set of labeled binary masks (see Figure 2.8, bottom-left):

$$\mathcal{B} = \{(\mathbf{M_j}, c_j)\}_{j=1}^{N}, \tag{2.16}$$

where each mask $\mathbf{M_j} \in \{0,1\}^{H \times W}$ indicates the presence of the j-th instance of class c_j.

The output of an instance segmentator is a set of predicted binary masks:

$$\hat{\mathcal{B}} = \{(\mathbf{M_i}, c_i)\}_{i=1}^{\hat{N}}. \tag{2.17}$$

The standard metric for instance segmentation is AP, and averaged across all classes mAP. The computation of AP is analogous to the object detection's AP.

Panoptic segmentation performs category-wise as well as instance-wise segmentation, distinguishing different "stuff" classes (e.g., road, sidewalk, terrain) from "things" classes (e.g., car, truck, tree, pole, person). In other words, it assigns a semantic label to each pixel of $\mathbf{I}$ or point of $\mathcal{P}$, then pixels or points that were labeled with a "thing" semantic label also are assigned an instance ID. Ground truth and output of a panoptic segmentator are analogous to an instance segmentator's ((see Figure 2.8, bottom-right)).

The standard metric for panoptic segmentation is panoptic quality (PQ). To compute PQ, we need to define three sets: true positive (TP), false positive (FP), and false negative (FN). They are computed per-class. Hence, for a class $c \in Y$:

$$\begin{aligned} TP_c &= \left\{ (A,B) \mid (A,c) \in \hat{\mathcal{B}}, (B,c) \in \mathcal{B}, \mathrm{IoU}(A,B) > \tau \right\} \\ FP_c &= \left\{ A \mid (A,c) \in \hat{\mathcal{B}}, (A,c) \notin TP_c \right\} \\ FN_c &= \left\{ B \mid (B,c) \in \mathcal{B}, (B,c) \notin TP_c \right\} \end{aligned} \tag{2.18}$$

where $\mathrm{IoU}(\cdot,\cdot)$ is computed as in semantic segmentation and τ is usually 0.5.

The per-class PQ_c is computed as:

$$\mathrm{PQ}_c = \frac{\sum_{(A,B)\in TP_c} \mathrm{IoU}(A,B)}{|TP_c| + \frac{1}{2}|FP_c| + \frac{1}{2}|FN_c|}. \tag{2.19}$$

PQ_c can be decomposed into segmentation quality (SQ_c) and recognition quality (RQ_c):

$$\mathrm{PQ}_c = \underbrace{\frac{\sum_{(A,B)\in TP_c} \mathrm{IoU}(A,B)}{|TP_c|}}_{\text{segmentation quality}} \cdot \underbrace{\frac{|TP_c|}{|TP_c| + \frac{1}{2}|FP_c| + \frac{1}{2}|FN_c|}}_{\text{recognition quality}}. \tag{2.20}$$

2.2.4 Place Recognition

Place recognition, also known as visual localization or image-based localization, is the task of determining a robot's location within a known environment using sensory data. It is a crucial component for autonomous navigation, enabling robots to recognize previously visited places and localize themselves on a pre-built map. Place recognition can be formulated as a classification problem, where the goal is to assign an input image to one of several predefined place categories. Formally, given an input image $\mathbf{I} \in \mathbb{R}^{H\times W\times C}$, the objective is to predict a place label $y \in Y = \{1,\ldots,P\}$, where P is the number of distinct places in the environment. Another formulation of the problem is image retrieval, where the goal is to find the most similar images from a database of known places.

A place recognition model is typically trained using a dataset of images captured from various locations within the environment, each annotated with the corresponding place label. The model learns to extract discriminative features from the images that are robust to changes in viewpoint, lighting, and occlusions.

During inference, let $\mathbf{I}_{\mathrm{db}}^{(i)} \in \mathbb{R}^{H\times W\times C}$ denote the i-th image in the database. The model extracts feature embeddings $\mathbf{f}_{\mathrm{query}} \in \mathbb{R}^D$ and $\mathbf{f}_{\mathrm{db}}^{(i)} \in \mathbb{R}^D$ using a feature extractor f_θ, parameterized by θ:

$$\mathbf{f}_{\mathrm{query}} = f_\theta(\mathbf{I}_{\mathrm{query}}), \quad \mathbf{f}_{\mathrm{db}}^{(i)} = f_\theta(\mathbf{I}_{\mathrm{db}}^{(i)}).$$

The similarity between the query image and the i-th database image is computed using a similarity metric $s(\cdot,\cdot)$, such as cosine similarity:

$$s(\mathbf{f}_{\mathrm{query}}, \mathbf{f}_{\mathrm{db}}^{(i)}) = \frac{\mathbf{f}_{\mathrm{query}} \cdot \mathbf{f}_{\mathrm{db}}^{(i)}}{\|\mathbf{f}_{\mathrm{query}}\| \|\mathbf{f}_{\mathrm{db}}^{(i)}\|},$$

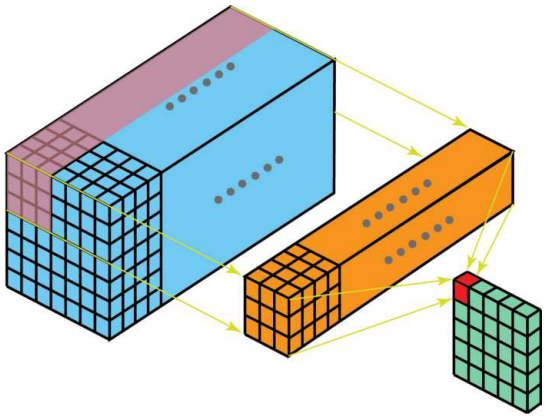


FIGURE 2.9: Visualization of a convolutional layer, showing the input tensor (blue), a convolutional filter (orange), and the resulting output tensor (green). The red cell highlights the initial feature value produced by the convolution operation.

or inverse Euclidean distance:

$$s(\mathbf{f}_{\text{query}}, \mathbf{f}_{\text{db}}^{(i)}) = -\|\mathbf{f}_{\text{query}} - \mathbf{f}_{\text{db}}^{(i)}\|_2.$$

The predicted place label $\hat{y}$ corresponds to the label of the database image with the highest similarity:

$$\hat{y} = y_{\text{db}}^{(i^*)}, \quad \text{where } i^* = \underset{i}{\text{argmax}}\, s(\mathbf{f}_{\text{query}}, \mathbf{f}_{\text{db}}^{(i)}).$$

The performance of place recognition systems is commonly evaluated using metrics such as accuracy, precision, recall, and the F1 score, similar to those used in image classification tasks.

2.3 Deep Learning-based 2D Vision Models

Deep learning has revolutionized the field of computer vision, enabling significant advancements in tasks such as image classification, object detection, and semantic segmentation. At the core of these breakthroughs are neural network architectures that can learn hierarchical feature representations from raw data, eliminating the need for hand-crafted features. This section introduces the foundational models and techniques that underpin modern vision-based perception systems, with a focus on their application to robotic perception.

We begin by exploring convolutional neural networks (CNNs), which have been the cornerstone of image-based deep learning. CNNs excel at capturing local spatial patterns and have been extended to dense prediction tasks such as semantic segmentation through fully convolutional networks (FCNs) and encoder-decoder architectures like U-Net and SegNet.

Building on the success of CNNs, attention mechanisms and transformer-based architectures have emerged as powerful tools for modeling long-range dependencies and global context. Vision transformers (ViTs) and their hierarchical variants, such as Pyramid Vision Transformers (PVTs), have demonstrated state-of-the-art performance in various vision tasks, offering a complementary approach to traditional convolutional methods.

2.3.1 Convolutional Neural Networks-based

The cornerstone of deep learning-based image processing, and in particular for 2D semantic segmentation, is the convolutional neural network (CNN) [32], which extracts hierarchical features from images through successive convolutional and pooling layers. Early CNNs

(e.g., AlexNet [91], VGG [170], ResNet [63]) designed for image classification were later extended to handle dense prediction tasks, leading to the development of fully convolutional networks (FCNs) [166, 16], which replace fully connected layers with convolutional ones to produce spatially consistent outputs.

A convolutional layer, illustrated in Figure 2.9 applies a set of learnable filters $W^{(l)} \in \mathbb{R}^{k_h \times k_w \times C_{in} \times C_{out}}$, where $k_h \times k_w$ is the filter size, to an input feature map $X^{(l)} \in \mathbb{R}^{H \times W \times C_{in}}$, producing output feature maps:

$$X^{(l+1)}_{h,w,k} = \sigma\left(\sum_{i=1}^{k_h} \sum_{j=1}^{k_w} \sum_{u=1}^{C_{in}} W^{(l)}_{i,j,u,c} \cdot X^{(l)}_{h+i,w+j,u} + b^{(l)}_c \right), \tag{2.21}$$

where $\sigma(\cdot)$ is a nonlinear activation function and $b^{(l)}_c \in \mathbb{R}$ is the bias term. Stacking multiple layers allows hierarchical feature extraction, from edges and textures to high-level semantics. More advanced convolutional layers also consider stride, padding, dilation, and grouped convolution for improving efficiency.

Opposite to the regular convolution, transposed convolution, also known as deconvolutions or fractionally strided convolutions, are used to upsample feature maps, effectively reversing the spatial reduction caused by standard convolutions and pooling operations. It is commonly applied for dense prediction tasks where the output must match the input resolution. The transposed convolution effectively "spreads" the input feature map by inserting zeros between elements (depending on the stride) and then applies the convolution operation. This results in an upsampled output with a larger spatial resolution.

A major advancement in segmentation has been the introduction of encoder–decoder architectures. The encoder progressively reduces the spatial resolution while learning high-level semantic features, and the decoder reconstructs the spatial resolution, producing pixel-wise class predictions. Architectures such as U-Net [157] and SegNet [5] introduced skip connections and refined upsampling techniques to preserve fine-grained details, which are crucial for precise segmentation.

U-Net

The U-Net architecture [157] is a fully CNN specifically designed for dense pixel-wise prediction tasks in image semantic segmentation. The network (see Figure 2.10) follows an encoder-decoder structure with a distinctive symmetric U-shaped topology, where the contracting path (encoder) systematically reduces spatial dimensions through repeated application of convolution operations followed by ReLU and max-pooling layers with stride 2, effectively capturing contextual information at multiple scales. The expanding path (decoder) employs transposed convolution operations to progressively restore spatial resolution while reducing feature channel dimensions. The architecture's defining characteristic lies in its skip connections, which concatenate feature maps from corresponding encoder layers directly to decoder layers at matching spatial resolutions, preserving fine-grained spatial information that would otherwise be lost during the downsampling process. The concatenations are at the cost of higher memory usage. Each level of the network typically doubles the number of feature channels while halving spatial dimensions during encoding, and reverses this process during decoding. The final layer applies a 1×1 convolution to map the multi-channel feature representation to the desired number of output classes.

SegNet

The SegNet architecture [5] is a CNN designed for dense pixel-wise prediction tasks in image semantic segmentation (see Figure 2.10). Also SegNet follows an encoder-decoder structure,

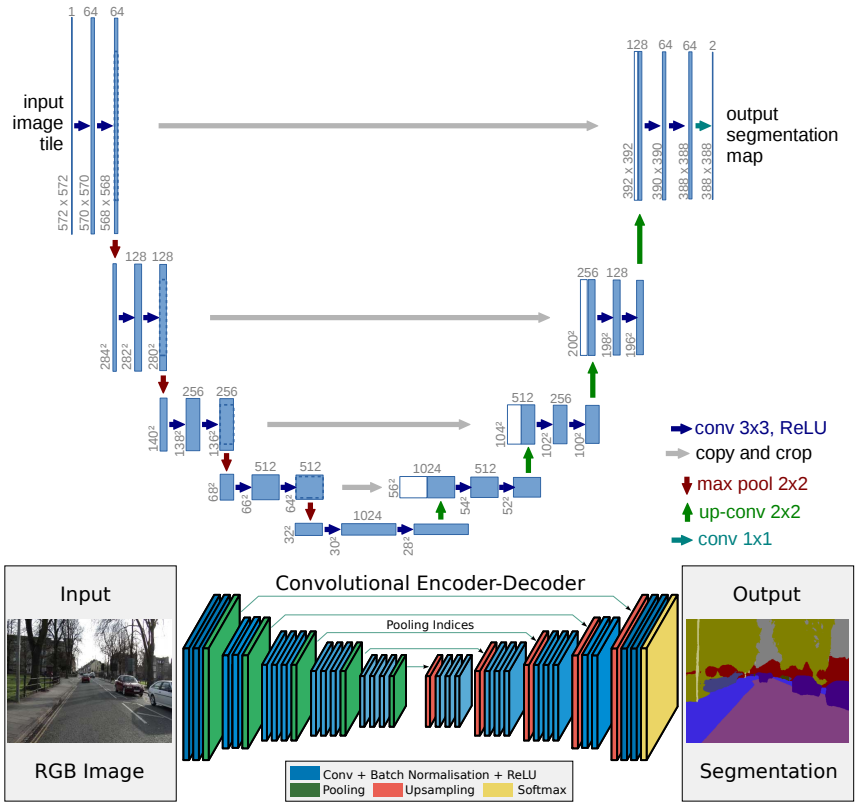


FIGURE 2.10: U-Net [157] (top) and SegNet [5] architectures: both follow an encoder-decoder structure, but U-Net uses skip connections to concatenate feature maps from the encoder to the decoder, while SegNet uses pooling indices for upsampling.

similarly to U-Net. The encoder consists of convolutional layers followed by batch normalization and max-pooling operations, which progressively reduce the spatial resolution while capturing high-level features The decoder mirrors the encoder structure but replaces max-pooling with unpooling layers, which use the pooling indices from the encoder to upsample the feature maps. This approach ensures that spatial information is preserved during upsampling, leading to more accurate segmentation boundaries. Unlike U-Net, SegNet does not use skip connections, which makes it more lightweight but potentially less effective at retaining fine-grained details.

Further refinements have included the integration of dilated (atrous) convolutions and multi-scale context aggregation, as exemplified in the DeepLabV3+ [21] family of models, which improved accuracy by capturing larger receptive fields without sacrificing resolution.

DeepLabV3+

DeepLabv3+ [21] is a semantic segmentation encoder-decoder architecture that combines atrous spatial pyramid pooling with depthwise separable convolutions to achieve state-of-the-art performance in semantic segmentation tasks.

Figure 2.11 depicts the DeepLabV3+ architecture. The encoder is composed by a CNN, typically ResNet-101 [63] or Xception [25], modified with atrous convolution, and the atrous spatial pyramid pooling (ASPP). Given a feature map $\mathbf{F} \in \mathbb{R}^{H' \times W' \times D}$, where H' and W' denote the spatial resolution and D the number of channels, the atrous (or dilated) convolution

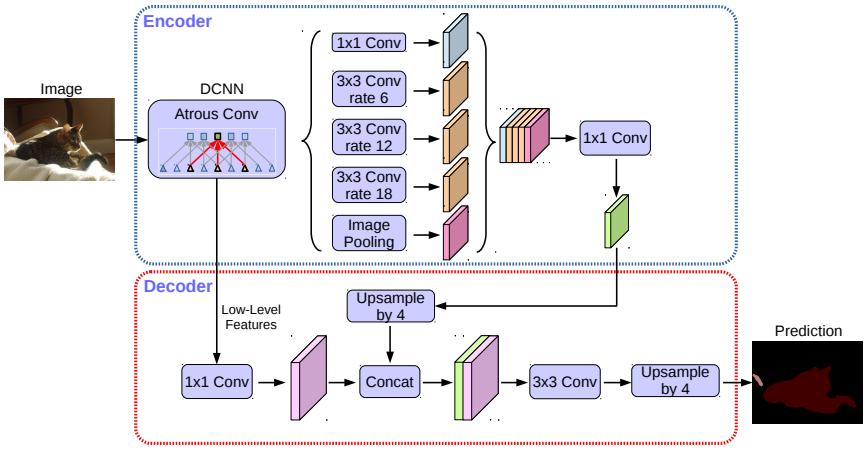


FIGURE 2.11: DeepLabv3+ [21] architecture: an encoder-decoder structure with atrous spatial pyramid pooling (ASPP) for multi-scale context aggregation and depthwise separable convolutions for efficient feature extraction.

with dilation rate $d \in \mathbb{N}$ and kernel $\mathbf{W} \in \mathbb{R}^{K\times K\times D\times D'}$ is defined as

$$(\mathbf{F} *_d \mathbf{W})(x) \;=\; \sum_u \mathbf{F}(x + d\cdot u)\,\mathbf{W}(u), \tag{2.22}$$

where:

- $x \in \mathbb{Z}^2$ indexes the spatial location in the output feature map,
- d is the dilation rate, controlling the stride with which input positions are sampled,
- $u \in \mathbb{Z}^2$ iterates over valid kernel positions,
- $\mathbf{F}(x + d\cdot u)$ is the input feature value at the dilated coordinate,
- $\mathbf{W}(u)$ is the learnable kernel weight associated with offset u.

Atrous convolution generalizes the standard convolution by inserting "holes" of size $d-1$ between sampled positions, expanding the receptive field without increasing the number of parameters or the amount of computation. It has the property of not modifying the spatial dimennsions H' and W'.

After the initial CNN, the ASPP module captures multi-scale context by applying parallel atrous convolutions with different dilation rates. It produces a concatenated tensor of features at different scales. A 1×1 convolution reduces the number of channels to match the output of the CNN. The outputs of CNN (low-level features) and ASPP (high-level features) form the input to the decoder module. The decoder upsamples and concatenates the low-level and ASPP features. Finally, a series of 3×3 convolutions refines the merged features, and bilinear upsampling restores the resolution to match the input image size.

2.3.2 Attention-based

In recent years, attention mechanisms have fundamentally reshaped the field of computer vision. Traditional CNNs have long been the dominant paradigm for visual understanding, owing to their ability to exploit local spatial correlations and translation invariance. However, their receptive fields grow only gradually through stacked layers, making it difficult to model long-range dependencies and global context. For complex vision tasks, capturing relationships between distant regions of an image is critical.

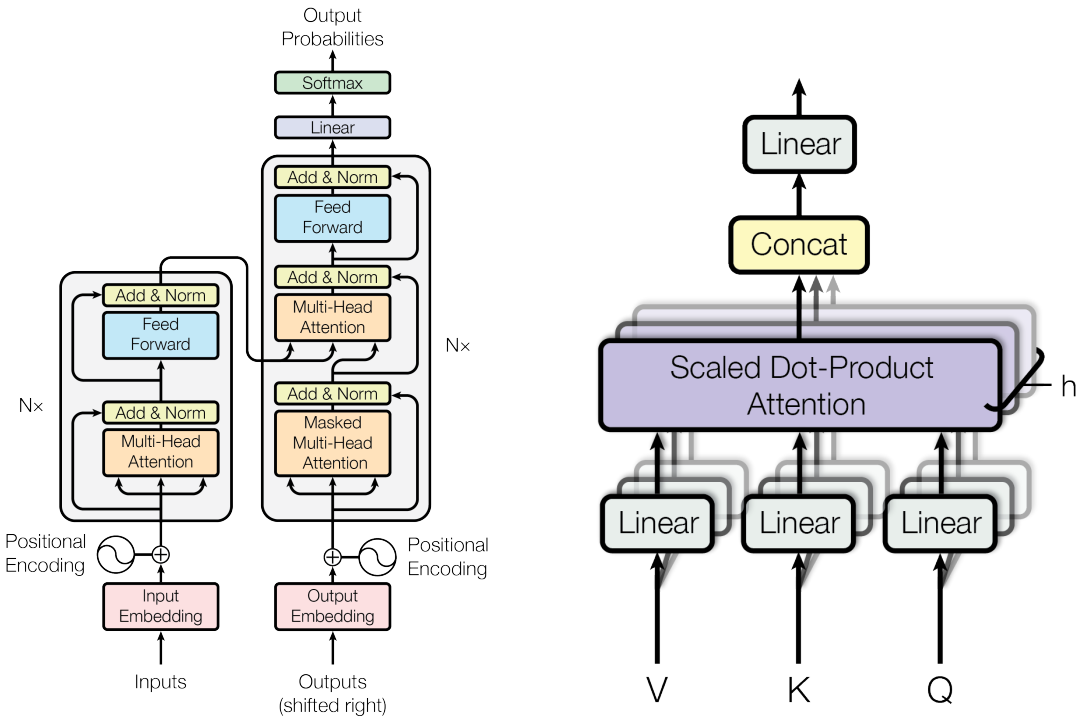


FIGURE 2.12: Left: the transformer architecture [190]. Right: the attention mechanism.

The introduction of attention mechanisms, first popularized in natural language processing (NLP) through the transformer architecture, provided a new way to overcome these limitations. The key idea of attention is to dynamically weight the contribution of different input elements depending on their relevance to a given query. Figure 2.12 illustrates the transformer architecture and the attention mechanism.

Consider an input sequence of N elements, each represented as a D-dimensional vector. We denote this input as

$$\mathbf{X} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^{N \times D} \tag{2.23}$$

Since the model has no inherent notion of sequence order, additional positional encodings $\mathbf{P} \in \mathbb{R}^{N \times D}$ are added to X, yielding $Z^{(0)} = \mathbf{X} + \mathbf{P}$. This representation $Z^{(0)}$ is then passed through a stack of L identical layers.

Given an input $Z \in \mathbb{R}^{N \times D}$, each element is projected into three different spaces through learned matrices:

Query: $\mathbf{Q} = \mathbf{Z}\mathbf{W}_Q, \quad \mathbf{W}_Q \in \mathbb{R}^{D \times D_k}$
Key: $\mathbf{K} = \mathbf{Z}\mathbf{W}_K, \quad \mathbf{W}_K \in \mathbb{R}^{D \times D_k}$
Value: $\mathbf{V} = \mathbf{Z}\mathbf{W}_V, \quad \mathbf{W}_V \in \mathbb{R}^{D \times D_v}$,

with $\mathbf{W}^Q, \mathbf{W}^K, \mathbf{W}^V \in \mathbb{R}^{d \times d_k}$ as trainable parameters. The attention operation is then defined as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D_k}}\right)\mathbf{V}, \tag{2.24}$$

where $\sqrt{D_k}$ is a scaling factor to stabilize gradients. The term $\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{D_k}} \in \mathbb{R}^{N \times N}$ represents pairwise similarity scores between all elements, which after the softmax are used as weights for a weighted sum of the values $\mathbf{V}$.

To capture dependencies from multiple representation subspaces, h attention heads are computed in parallel:

$$\text{head}_i = \text{Attention}(\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i), \tag{2.25}$$

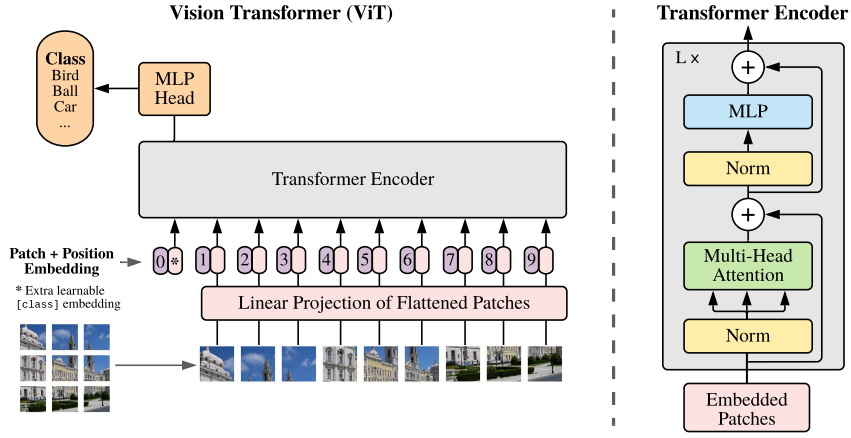


FIGURE 2.13: ViT architecture [37]. An input image is divided into patches, which are linearly embedded, combined with positional encodings, and processed through a series of transformer encoder layers to produce a final representation for classification.

where $\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i$ are projections of $\mathbf{Z}$ with separate trainable matrices for each head.

The outputs of all heads are concatenated and linearly transformed:

$$\text{MultiHead}(\mathbf{Z}) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h)\mathbf{W}_O, \tag{2.26}$$

with $\mathbf{W}_O \in \mathbb{R}^{(hD_v)\times D}$ as a trainable projection matrix.

Each encoder layer consists of a multi-head self-attention module followed by a position-wise feedforward network (FFN), with residual connections and layer normalization applied after each sub-layer:

$$\begin{aligned} \mathbf{Z}' &= \text{LayerNorm}(\mathbf{Z} + \text{MultiHead}(\mathbf{Z})) \\ \mathbf{Z}^{(out)} &= \text{LayerNorm}(\mathbf{Z}' + \text{FFN}(\mathbf{Z}')), \end{aligned} \tag{2.27}$$

where the FFN consists of two linear transformations with a ReLU activation in between:

$$\text{FFN}(\mathbf{Z}) = \text{ReLU}(\mathbf{Z}\mathbf{W}_1 + b_1)\mathbf{W}_2 + b_2, \tag{2.28}$$

with $\mathbf{W}_1 \in \mathbb{R}^{D\times D_{ff}}, \mathbf{W}_2 \in \mathbb{R}^{D_{ff}\times D}$ and b_1, b_2 as trainable parameters.

Stacking L such layers produces the final encoder representation $\mathbf{Z}^{(L)}$, which can then be used for downstream tasks.

The transformer architecture consists also of a decoder, which takes the encoder output and previous target elements to generate the next element in a sequence.

The attention mechanism has a complexity of $O(N^2D)$, which can be computationally expensive for large N. However, its ability to model global dependencies and context has led to significant performance improvements across various tasks.

In vision, this means that each region of an image can directly attend to other regions, regardless of their spatial distance, enabling a more holistic representation of the scene. The most influential instantiation of this concept is the vision transformer.

Vision-Transformers

The vision transformer (ViT) [37] is a pioneering architecture that adapts the transformer model to image recognition tasks. As illustrated in Figure 2.13, ViT divides an input image $\mathbf{I} \in \mathbb{R}^{H\times W\times C}$ into a grid of non-overlapping patches, each of size $P \times P$, with both H and W divisible by P. Each patch is then flattened and linearly projected into a D-dimensional

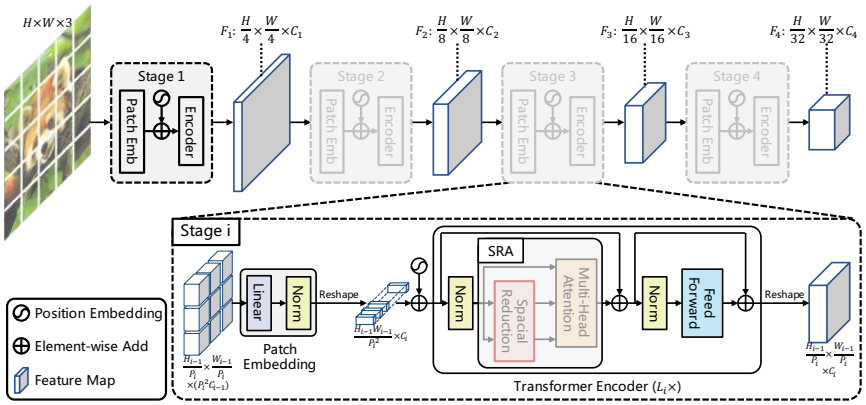


FIGURE 2.14: PVT architecture [201]. The model processes an input image through multiple stages, each reducing spatial dimensions while increasing feature channels, using spatial reduction attention to manage computational complexity.

embedding space, resulting in a sequence of patch embeddings:

$$\mathbf{X} = [x_1, \ldots, x_N]^\top \in \mathbb{R}^{N \times D}, \tag{2.29}$$

where $N = \frac{HW}{P^2}$ is the number of patches.

To retain positional information, learnable positional encodings $\mathbf{P} \in \mathbb{R}^{N \times D}$ are added to the patch embeddings, yielding:

$$Z^{(0)} = \mathbf{X} + \mathbf{P}. \tag{2.30}$$

This sequence $Z^{(0)}$ is then processed through a stack of L transformer encoder layers.

ViT has demonstrated remarkable performance in image classification tasks, achieving state-of-the-art results on benchmarks such as ImageNet [34]. However, its reliance on large-scale datasets for pretraining remains a limitation, as smaller datasets may lead to overfitting.

Pyramid Vision Transformers

In addition to the ViT, the pyramid vision transformer (PVT) [201] has extended the transformer paradigm to dense prediction tasks, such as semantic segmentation and object detection. By incorporating a hierarchical structure and spatial reduction attention, PVT achieves a balance between computational efficiency and performance, making it suitable for real-time applications in robotics. As shown in Figure 2.14, PVT follows a multi-stage architecture, where each stage consists of several transformer encoder layers. Each stage reduces the spatial dimensions of the feature maps while increasing the number of channels, similar to traditional CNNs. To address the quadratic complexity of standard self-attention, PVT introduces spatial reduction attention (SRA), which reduces the spatial dimensions of the key and value matrices before computing attention. This modification significantly lowers the computational burden, allowing the model to handle higher-resolution inputs efficiently.

PVTv2 [202] further refines the architecture by introducing overlapping patch embedding and a convolutional feedforward network, enhancing the model's ability to capture local details while maintaining global context.

2.4 Deep Learning-based 3D Vision Models

This section explores the foundational models and techniques for 3D vision, focusing on 3D point cloud semantic segmentation methods. As discussed in Section 2.2.3, semantic segmentation of point clouds assigns a semantic label to each point, enabling detailed scene

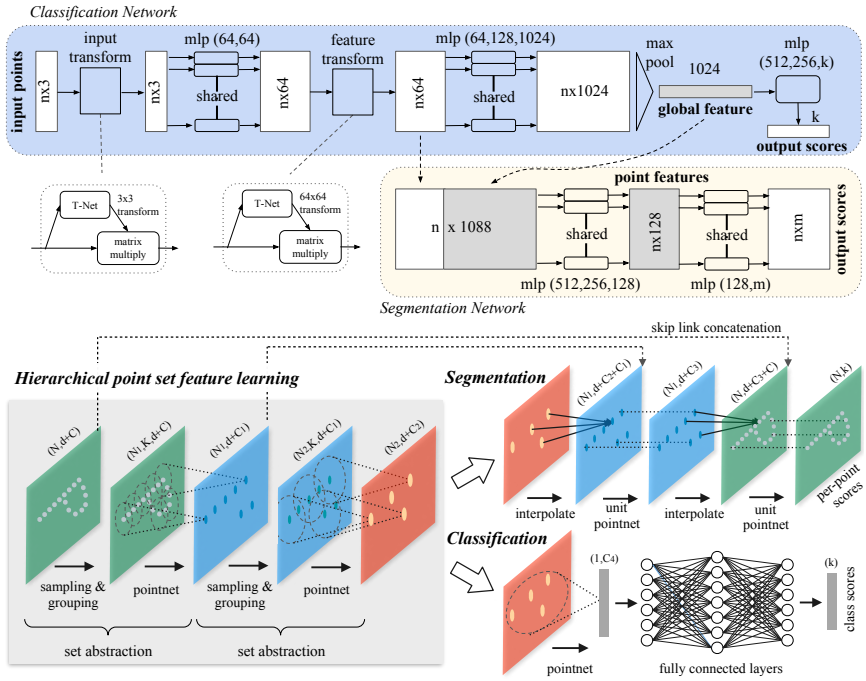


FIGURE 2.15: PointNet [147] (top) and PointNet++ [148] (bottom) architectures: PointNet processes each point independently with shared MLPs and aggregates features with max-pooling, while PointNet++ introduces hierarchical grouping and local feature learning to capture multi-scale context.

understanding in three dimensions. Different approaches have been developed to address the unique challenges, such as irregularity, sparsity, and varying density, of point cloud data:

- point-based;
- voxel/grid-based;
- projection-based;
- cluster-based;
- hybrid methods.

Each class of methods offers a trade-off between computational efficiency, geometric accuracy, and robustness to varying point cloud density.

Compared to image-based segmentation, 3D semantic segmentation provides richer geometric information, which is critical for tasks such as traversability analysis, obstacle avoidance, and mapping. However, it lacks the color and texture cues available in 2D images, and annotated 3D datasets are typically smaller and harder to obtain. As a result, 3D segmentation methods often face greater data scarcity and computational challenges. In practice, many robotic systems combine 2D and 3D modalities, leveraging the complementary strengths of both to achieve more robust perception.

Following is a brief overview of some representative methods from each category.

2.4.1 Point-based methods

Point-based methods operate directly on raw point clouds, learning features from the unordered set of points. Models such as PointNet and its successors demonstrated that permutation-invariant architectures could effectively capture both local and global geometric structures.

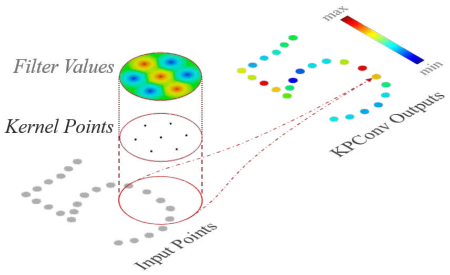


FIGURE 2.16: KPConv [186]: kernel point convolution operation defines convolutional kernels as learnable points in 3D space, interacting with nearby input points.

PointNet

PointNet [147] is the first deep learning model to process point sets directly. The architecture is depicted in Figure 2.15 It applies a shared MLP to each point independently, producing point-wise embeddings. To achieve permutation invariance, a symmetric function (typically max-pooling) aggregates point features into a global descriptor. This global feature can then be concatenated back with point-level features for dense prediction tasks such as semantic segmentation. This network introduced the key idea of permutation invariance via symmetric pooling.

PointNet++

PointNet++ [148] extends PointNet with a hierarchical feature extraction scheme. Using farthest point sampling (FPS) and ball query grouping, the network (see Figure 2.15) organizes the point cloud into neighborhoods at multiple scales. Within each neighborhood, a PointNet-like MLP processes local regions. This hierarchy resembles the multi-scale receptive fields of 2D CNNs, enabling the model to capture both fine details and global context.

KPConv

PointNet++ still relies on MLPs inside neighborhoods, which limits geometric expressiveness. KPConv [186] introduces kernel point convolutions (see Figure 2.16), which define convolutional kernels as a set of learnable points in 3D space. Each kernel point interacts with nearby input points via distance-weighted contributions, allowing the network to explicitly model geometric structures. Both rigid (fixed kernel points) and deformable (learnable kernel point positions) variants exist, with the deformable version adapting kernel shape to local geometry.

RandLA-Net

Point-based methods often scale poorly to large outdoor point clouds due to the quadratic cost of neighborhood queries. RandLA-Net [69] (see Figure 2.17) addresses scalability by introducing random sampling for downsampling and local feature aggregation for context modeling. Instead of expensive FPS, it randomly selects subsets of points, drastically improving efficiency. To preserve representational power, an attentive pooling mechanism aggregates local context, ensuring that even randomly sampled subsets retain key geometric features. RandLA-Net achieves real-time performance on large-scale LiDAR datasets (e.g., SemanticKITTI [6]), making point-based methods practical for robotics applications.

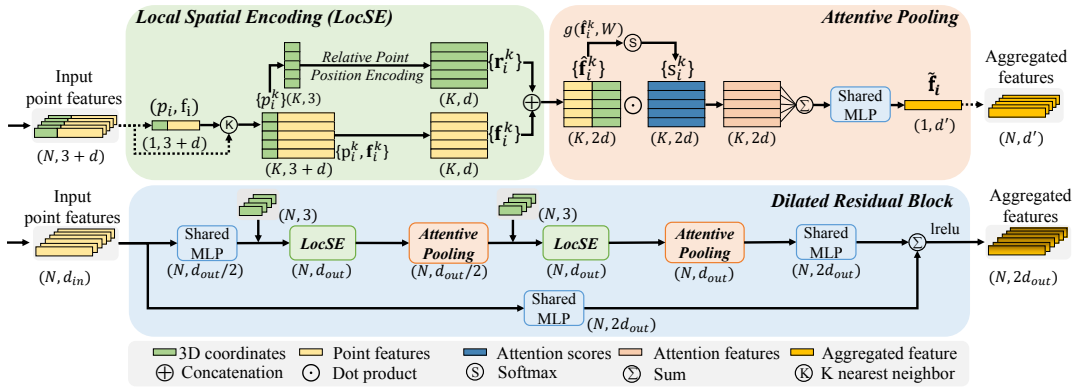


FIGURE 2.17: RandLA-Net [69] architecture: it uses random sampling and local feature aggregation for efficient large-scale point cloud processing.

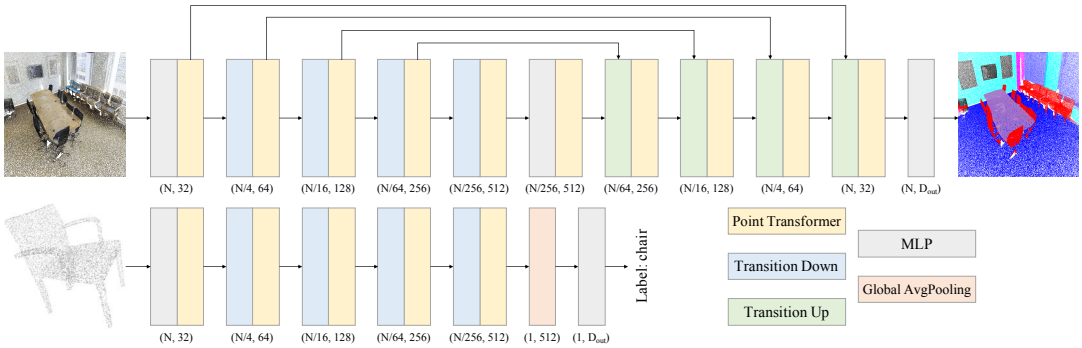


FIGURE 2.18: Point Transformer [238] architecture: it employs self-attention mechanisms to capture long-range dependencies in point clouds.

Point Transformer series

Point transformer v1 (PTv1) [238] (see Figure 2.18) adapts the transformer architecture to point clouds. Long-range dependencies are difficult to model with MLPs or local convolutions. For each point, the network computes relationships with its neighbors through self-attention, guided by learned position encodings (called conditional position encoding (CPE)) that account for relative 3D coordinates. It's important to note that self-attention is applied locally. For each point, PTv1 finds its K nearest neighbors in Euclidean space. Then, self-attention is applied within this local neighborhood, not globally. With KNN, the network achieves points permutation invariance. The architecture follows an encoder-decoder, U-Net paradigm, where the number of points gets shrinked and feature size increments in the encoder part, and vice versa in the decoder. PTv1 was designed and tested only on indoor datasets. As the authors of PTv1 note in [238], as the model goes deeper and the number of channels increases, the number of weight encoding parameters also increases drastically, leading to severe overfitting and limiting the model depth.

Point transformer v2 (PTv2) [215] addresses the limitations of PTv1 by introducing grouped vector attention, featuring a more parameter-efficient formulation, where the vector attention is divided into groups with shared vector attention weights. In addition, PTv2 adopts a different positional encoding strategy, inserting a multiplicative, MLP-produced spatial embedding, before adding the (common) additive spatial information. In general, the U-Net architecture with skip-connection is preserved from PTv1, and also the local KNN-based local attention. Still, PTv2 is only tested in indoor datasets.

Point transformer v3 (PTv3) [214] (see Figure 2.19) is a game changer in the point

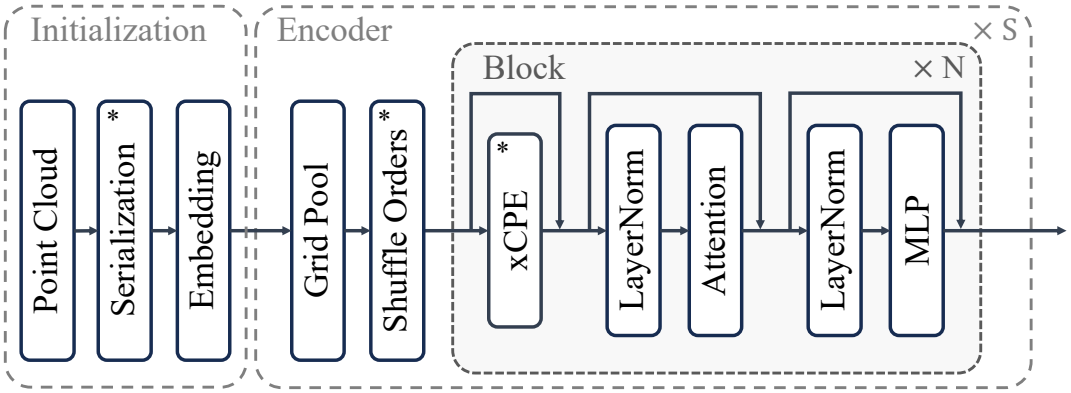


FIGURE 2.19: Point Transformer v3 [214] architecture: it introduces point cloud serialization, patch-attention and xCPE to reach state-of-the-art performance both in indoor and outdoor environments.

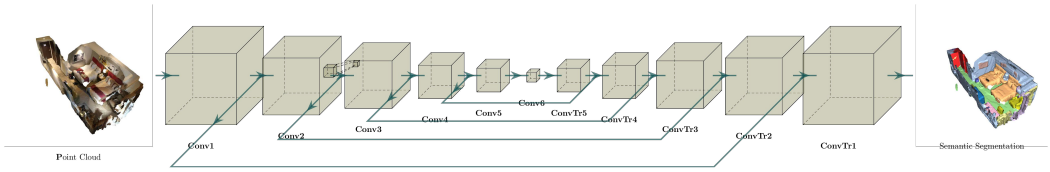


FIGURE 2.20: MinkowskiNet [26] architecture: a UNet-like encoder-decoder structure with sparse 3D convolutions and pooling operations for efficient processing of large-scale 3D point clouds.

transformer series, prioritizing simplicity and efficiency over the accuracy of certain mechanisms (e.g., spatial proximity definition), thereby enabling scalability. In detail (see the architecture in Figure 2.19), it introduces point cloud serialization to avoid the slow KNN operations. Unordered, unstructured point clouds are first voxelized, then points are ordered according to the so-called space-filling curves [144]. The ordering effectively rearranges the points in such a way that neighbor points in the data structure are also likely to be close in space. Then, points are grouped into non-overlapping patches, based on their ordering, and patch-attention performs attention within each individual patch. PTv3 also improves CPE to extended CPE (xCPE), making it compatible with serialized point clouds. While the serialization strategy may temporarily yield a lower performance than some neighborhood construction strategies like KNN, authors demonstrate its efficacy. PTv3 is tested both on indoor datasets and outdoor datasets, reaching state-of-the-art performance in both setups.
The point transformer series demonstrates that attention-based architectures can model both local and global relationships in point clouds, achieving state-of-the-art results in 3D semantic segmentation.

2.4.2 Voxel/grid-based methods

Voxel/grid-based methods discretize space into a 3D grid of voxels, allowing the use of convolutional architectures similar to those in image analysis. While effective, this approach can be computationally expensive and memory-intensive due to the sparsity of point clouds.

Key to many voxel-based methods is 3D convolution, which extends the 2D convolution operation to three dimensions. Given a 3D input tensor $\mathbf{X} \in \mathbb{R}^{D \times H \times W \times C_{in}}$, where D, H, and W are the depth, height, and width dimensions respectively, and C_{in} is the number of input channels, a 3D convolutional layer applies a set of learnable 3D filters $\mathbf{W} \in \mathbb{R}^{k_d \times k_h \times k_w \times C_{in} \times C_{out}}$, where k_d, k_h, and k_w are the filter sizes in each dimension, to

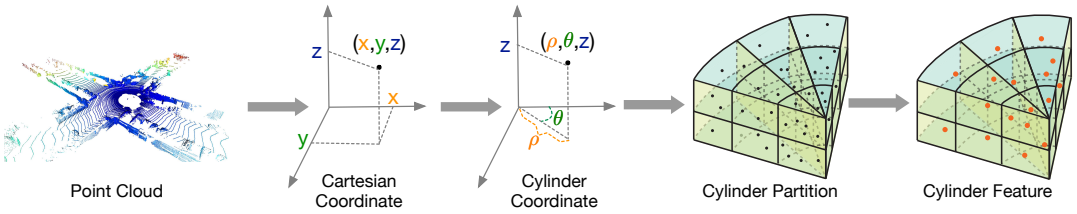


FIGURE 2.21: Cylinder3D [242] architecture: it uses cylindrical partitioning and sparse 3D convolutions for LiDAR point cloud semantic segmentation.

produce an output tensor $\mathbf{Y} \in \mathbb{R}^{D' \times H' \times W' \times C_{out}}$:

$$\mathbf{Y}^{(l+1)}_{d,h,w,c} = \sigma\left(\sum_{i=1}^{k_d} \sum_{j=1}^{k_h} \sum_{m=1}^{k_w} \sum_{u=1}^{C_{in}} \mathbf{W}^{(l)}_{i,j,m,u,c} \cdot \mathbf{X}^{(l)}_{d+i,h+j,w+m,u} + b^{(l)}_c \right), \tag{2.31}$$

where $\sigma(\cdot)$ is a nonlinear activation function and $b^{(l)}_c \in \mathbb{R}$ is the bias term.

MinkowskiNet and 3D Sparse Convolutions

Most real-world point clouds are extremely sparse: only a small fraction of voxels contain data. Dense 3D convolutions therefore waste vast computational resources. MinkowskiNet [26] generalized sparse convolutions to higher dimensions (space and time), enabling scalable training and inference on large LiDAR scenes. The architecture (see Figure 2.20) typically employs a UNet-like encoder–decoder backbone, but all convolutions and pooling operations are sparse, ensuring efficiency. MinkowskiNet is a milestone, showing that sparse convolutions enable voxel-based methods to scale to outdoor datasets like SemanticKITTI while maintaining high accuracy.

Cylinder3D

Standard cubic voxelization does not respect the distribution of LiDAR points, which are denser near the sensor and sparser farther away. Cylinder3D [242] proposes cylindrical partitioning of space instead of Cartesian grids. This improves sampling uniformity and representation power for LiDAR data. A sparse 3D CNN backbone processes the cylindrical voxels, capturing both geometric structures and long-range context (see Figure 2.21).

Point-Voxel Knowledge Distillation

Point-voxel knowledge distillation (PVKD) [67] is a hybrid architecture (see Figure 2.22) that combines the strengths of voxel-based and point-based methods for efficient and accurate 3D semantic segmentation. The model leverages a two-stream framework: a voxel-based teacher network and a point-based student network. The teacher network processes the input point cloud using sparse 3D convolutions to extract high-level semantic features, while the student network operates directly on raw points, learning fine-grained details. To bridge the two streams, PVKD employs a knowledge distillation mechanism, where the teacher network's voxel-level features are interpolated back to the point cloud and used to guide the student network's training. This enables the student network to inherit the teacher's global context understanding while maintaining the flexibility and efficiency of point-based processing. The architecture also incorporates a multi-scale feature aggregation strategy, combining features from different layers of the teacher network to enhance the student's ability to capture both local and global geometric structures. This hybrid approach achieves state-of-the-art performance on large-scale LiDAR datasets.

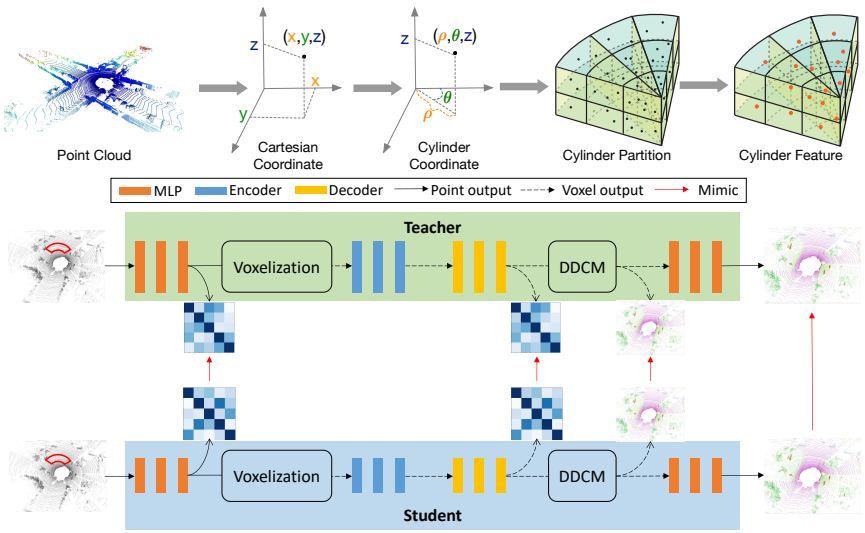


FIGURE 2.22: PVKD [67] architecture: it combines voxel-based and point-based processing with knowledge distillation for efficient 3D point cloud semantic segmentation.

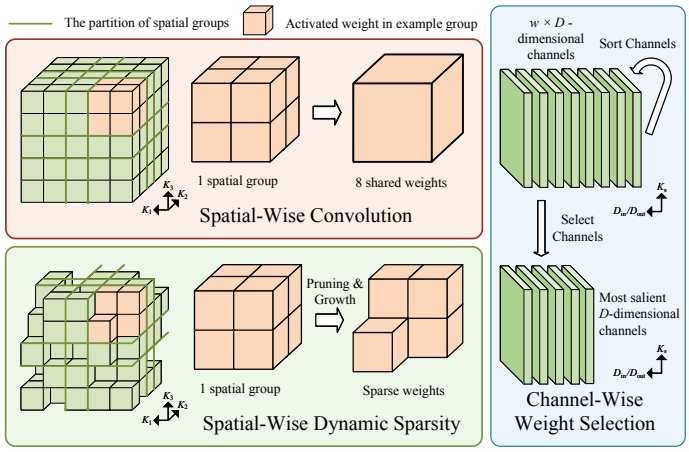


FIGURE 2.23: LSK3DNet [43] dynamic sparsity convolution and channel pruning mechanism for efficient large-kernel 3D convolutions.

LSK3DNet

Sparse convolution-based methods usually rely on stacking many small 3D kernels (e.g., $3 \times 3 \times 3$), which limits the effective receptive field and struggles to capture large-scale context, which is critical for semantic segmentation in complex outdoor environments. LSK3DNet [43] introduces large sparse kernels as the core building block. It is illustrated in Figure 2.23. Instead of relying on multiple layers of small kernels, LSK3DNet directly applies large convolutional kernels on sparse voxel grids, significantly enlarging the receptive field. To keep computation tractable, the method employs dynamic sparsity and channel pruning, learning during training which kernel elements to activate. This makes large-kernel operations both effective and efficient. The architecture is modular, combining hierarchical sparse convolution blocks with these optimized large-kernel operations.

LSK3DNet demonstrates that explicitly modeling large receptive fields with learned sparse structures provides state-of-the-art performance on large-scale 3D semantic segmentation benchmarks, especially for terrain-rich and cluttered environments.

2.4.3 Projection-based methods

Projection-based methods map 3D point clouds into 2D representations (e.g., spherical range images or bird's-eye views), enabling the reuse of established 2D convolutional techniques while trading off some geometric fidelity.

Key to many projection-based methods is the concept of a range image. A range image is a 2D representation of a 3D point cloud, where each pixel corresponds to a LiDAR measurement. It is constructed by projecting the 3D points onto a spherical coordinate system defined by the LiDAR sensor. Let a point in the 3D Cartesian coordinate system be represented as $(x, y, z) \in \mathbb{R}^3$. The corresponding spherical coordinates (r, θ, ϕ) are given by:

$$\begin{aligned} r &= \sqrt{x^2 + y^2 + z^2}, \\ \theta &= \arctan 2(y, x), \\ \phi &= \arcsin\left(\frac{z}{r}\right), \end{aligned} \tag{2.32}$$

where r is the radial distance, θ is the azimuth angle, and ϕ is the elevation angle.

The range image is a 2D grid where each pixel (u, v) corresponds to a specific azimuth and elevation angle. The mapping from spherical coordinates to pixel indices is defined as:

$$\begin{aligned} u &= \left\lfloor \frac{\theta - \theta_{\min}}{\Delta\theta} \right\rfloor, \\ v &= \left\lfloor \frac{\phi - \phi_{\min}}{\Delta\phi} \right\rfloor, \end{aligned} \tag{2.33}$$

where $\theta_{\min}$ and $\phi_{\min}$ are the minimum azimuth and elevation angles of the LiDAR sensor, and $\Delta\theta$ and $\Delta\phi$ are the angular resolutions in the azimuth and elevation directions, respectively.

Most modern LiDAR sensors encode the ring channel, which corresponds to the vertical laser channel index, directly into the data. This simplifies the elevation angle computation, as each ring is associated with a fixed elevation angle ϕ_i. The elevation angle can then be determined using a pre-calibrated lookup table, avoiding the need for explicit spherical coordinate transformations.

The value at each pixel (u, v) in the range image is the radial distance r of the corresponding 3D point:

$$\text{RangeImage}(u, v) = r. \tag{2.34}$$

If multiple points map to the same pixel, the point with the smallest r (closest to the sensor) is typically selected. This ensures that the range image captures the visible surface of the environment.

Range images are compact and structured, making them suitable for applying 2D convolutional neural networks while preserving the geometric information of the original 3D point cloud.

RangeNet++

LiDAR naturally provides range images, where each beam corresponds to a pixel in a 2D spherical projection. RangeNet++ [129] applies a 2D CNN to range images and then reprojects the pixel predictions back to 3D points. It uses a modified version of the Darknet-53 architecture, which is optimized for semantic segmentation tasks. This enables efficient processing with standard CNNs while maintaining the semantics of the original point cloud. RangeNet++ demonstrated the feasibility of 2D projection approaches, achieving real-time performance on SemanticKITTI. The architecture is depicted in Figure 2.24.

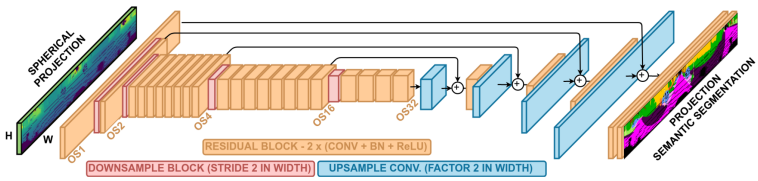


FIGURE 2.24: RangeNet++ [129] architecture: a 2D CNN processes range images, followed by a re-projection layer that maps 2D predictions back to the 3D point cloud for semantic segmentation.

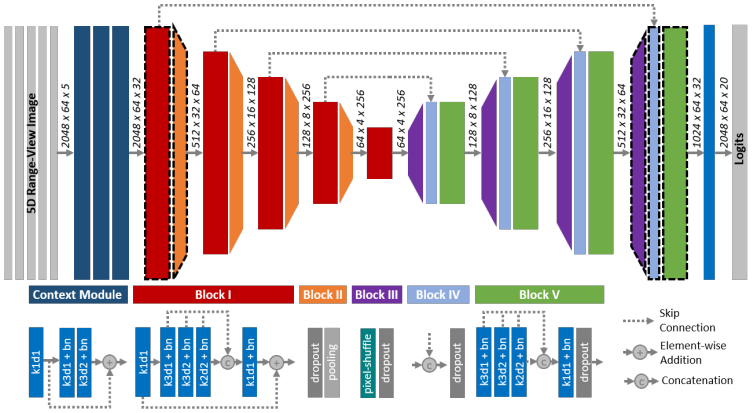


FIGURE 2.25: SalsaNext [30] architecture: an enhanced 2D CNN with squeeze-and-excitation modules and residual connections for improved range image segmentation.

SalsaNext

RangeNet++ sufferes from limited accuracy due to projection-induced ambiguities. SalsaNext [30] (see Figure 2.25) improves the encoder–decoder backbone with squeeze-and-excitation modules, residual connections, and context modules to capture larger receptive fields in the range image domain. SalsaNext improved the balance of efficiency and accuracy, setting a strong baseline for range-image methods.

CENet

CENet [23] further enhances range-image segmentation by introducing a context encoding module that captures global scene context (see Figure 2.26). This module aggregates features across the entire range image, allowing the network to leverage global information for more accurate predictions. CENet also incorporates a multi-scale feature fusion strategy, combining features from different layers of the network to improve segmentation performance. Auxiliary losses, both at the encoder and the decoder modules, helps the training to converge faster.

WaffleIron

Instead of using only the range image, WaffleIron [145] combines multiple 2D projections to capture complementary information. WaffleIron (see Figure 2.27) constructs a ”waffle” of projections, including range images, bird’s-eye view (i.e., XY projection), XZ and YZ projections. An initial point-wise point cloud embedding extracts per-point local features by processing neighborhood of points with MLPs and max-pooling. These features are then projected into the different 2D views, where separate 2D CNNs process each projection. A series of spatial mix and channel mix operations extracts point-wise features.

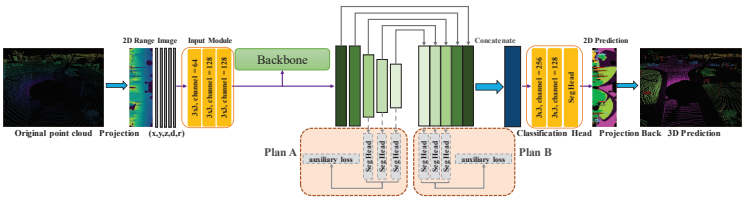


FIGURE 2.26: CENet [23] architecture: a pyramid CNN with auxiliary losses for range-image based point cloud semantic segment

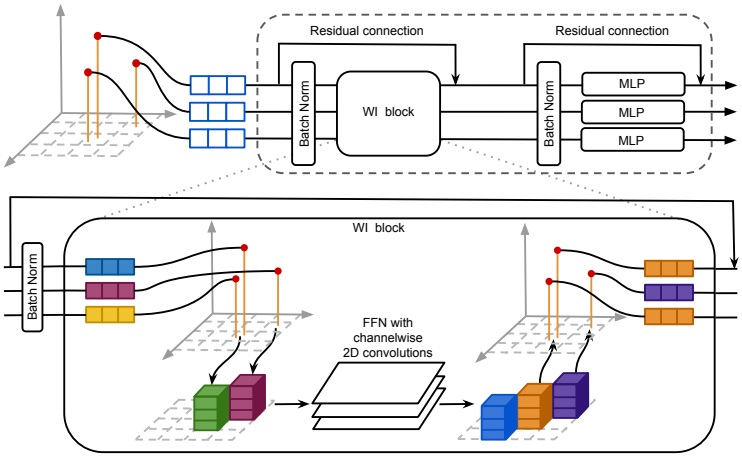


FIGURE 2.27: WaffleIron [145] architecture: a multi-view approach combining range images, BEVs, and other projections with point-wise feature extraction for 3D semantic segmentation.

This multi-view approach captures richer geometric context, improving segmentation accuracy while maintaining efficiency.

RangeFormer

CNN-based projection methods effectively exploit local structures in range images but they struggle to model long-range dependencies due to limited receptive fields. Rangeformer [90] integrates transformer-based attention mechanisms into the range image segmentation framework. The architecture (see Figure 2.28) is similar to PVTv2's backbone, using spatial reduction attention to manage computational complexity. Rangeformer achieves state-of-the-art performance on large-scale LiDAR datasets, demonstrating the effectiveness of attention mechanisms in projection-based segmentation.

2.4.4 Cluster-based methods

Cluster-based methods process a point cloud by using the super-point representation, in which points in the cloud are clustered and the clusters (called super-points) are then connected in a graph representation, which can be processed by graph convolutional neural networks (GCNNs).

SuperPoint Graph (SPG)

One of the earliest and most influential works in this direction is the Superpoint Graph (SPG) [94]. The point cloud is first partitioned into superpoints using unsupervised geometric clustering. Each superpoint is intended to be geometrically consistent (e.g., all points belonging to a flat patch of ground). A graph is constructed where nodes correspond to superpoints,

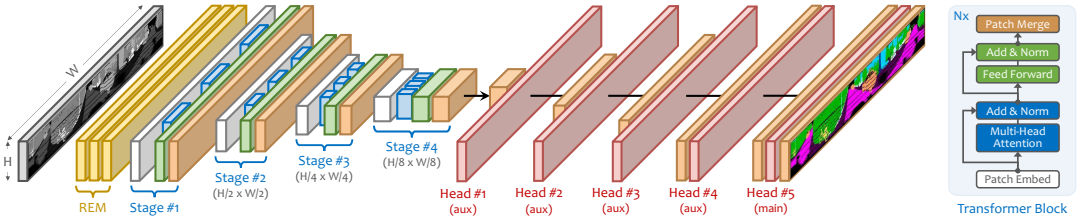


FIGURE 2.28: Rangeformer [90] architecture: a UNet-like encoder-decoder structure with transformer-based attention mechanisms for capturing long-range dependencies in range images.

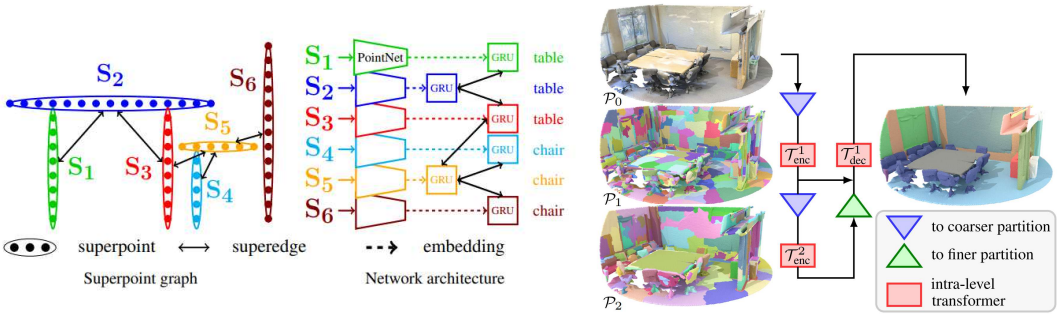


FIGURE 2.29: Superpoint Graph (SPG) [94] (left) and Superpoint Transformer (SPT) [153] (right) architectures. In SPG, a graph is built over superpoints and a PointNet module extracts per-cluster features, which are then refined by a G-GNN module. In SPT, a transformer-based model operating on superpoints, capturing long-range dependencies through attention for 3D semantic segmentation.

and edges capture adjacency relations (spatial proximity, geometric continuity). A PointNet module extracts per-cluster features. A gated-graph neural network (G-GNN) is applied on top of the graph, propagating contextual embedding information across clusters to classify each superpoint using gated recurrent unit (GRU) update rule during message passing.

SPG introduced the idea that 3D semantic segmentation benefits from mid-level structure. By reasoning on superpoints rather than raw points, it achieves strong efficiency and captures scene-level context more effectively. It was tested both in indoor (S3DIS [4]) and outdoor (Semantic3D [59]) datasets.

Superpoint Transformer (SPT)

The Superpoint Transformer (SPT) [153] (see Figure 2.29) extends SPG by replacing the GNN with transformer-based attention, which provides more flexible and long-range interactions between superpoints. It takes as input superpoints, i.e. clusters of points represented by features (geometry, color, learned embeddings). A transformer encoder operates on the set of superpoints, dynamically learning which regions should attend to each other, rather than relying only on fixed graph edges.

Superpoints are clusters of points that are geometrically consistent, and they are usually obtained through unsupervised geometric clustering. The authors of this paper also propose a new method to compute a hierarchical superpoint structure for large point clouds, which is more than 7 times faster than existing superpoint-based methods. The model takes as input superpoints represented by features (RGB/intensity, PCA-based, elevation, and other handcrafted features). The graph-like structure is processed by a graph-attention convolutional network that captures long-range dependencies between superpoints through attention mechanisms. Per-cluster semantic predictions are obtained via an MLP head which refines superpoint embeddings. SPT demonstrated significant improvements on large-scale datasets

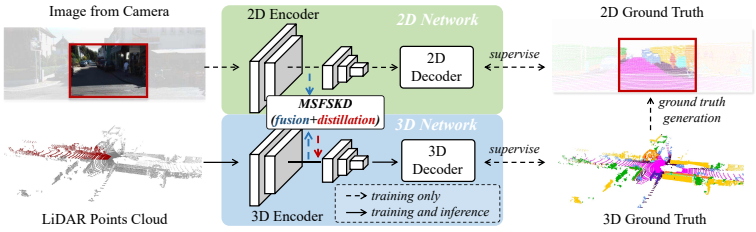


FIGURE 2.30: 2D-PASS [224] architecture: a hybrid model that fuses 2D image features from a pretrained segmentation network with 3D LiDAR features using attention for 3D semantic segmentation.

(e.g., indoor S3DIS [4] and outdoor Semantic3D [59]), and showed that attention mechanisms naturally complement cluster-based representations.

2.4.5 Hybrid methods

Hybrid methods fuse information from more than one sensor, e.g. fusing LiDAR and RGB images features. One of the most successful examples is 2DPASS.

2DPASS

2D priors assisted semantic segmentation (2DPASS) [224] explicitly leverages image priors to improve LiDAR semantic segmentation. The intuition is that pretrained 2D semantic segmentation models encode rich semantic knowledge that can complement sparse LiDAR geometry, especially for visually distinct classes such as vegetation, cars, and pedestrians.

The pipeline (illustrated in Figure 2.30) first projects LiDAR points onto the image plane of a calibrated camera. A pretrained 2D semantic segmentation network (e.g., DeepLabv3+) produces dense semantic logits on the image. These are then back-projected to the 3D space and fused with LiDAR features extracted by a 3D backbone (e.g., a sparse convolutional network). The fusion occurs through attention-based modules, allowing the network to learn how to weight 2D and 3D features for each point.

2D-PASS demonstrates the power of cross-modal distillation from vision to LiDAR. By transferring the semantic richness of large-scale 2D models into the 3D domain, it sets new benchmarks on SemanticKITTI and nuScenes, especially in categories where LiDAR alone struggles.

2.5 3D Instance/Panoptic Segmentation Methods

Point cloud instance and panoptic segmentation methods are becoming more prevalent in the recent years thanks to the advancement in deep learning. These approaches are inherently more challenging than semantic segmentation, as they require not only classifying each point but also distinguishing between different object instances. They can be broadly categorized into proposal-based methods, which generate object proposals and refine them into instance masks, and proposal-free methods, which directly predict instance labels for each point. Proposal-based methods are Mask3D, MaskPLS, OneFormer3D and Spherical Mask, while proposal-free methods include two methods similarly named, i.e. Superpoint Transformer (SPT) and Superpoint Transformer (SPFormer).

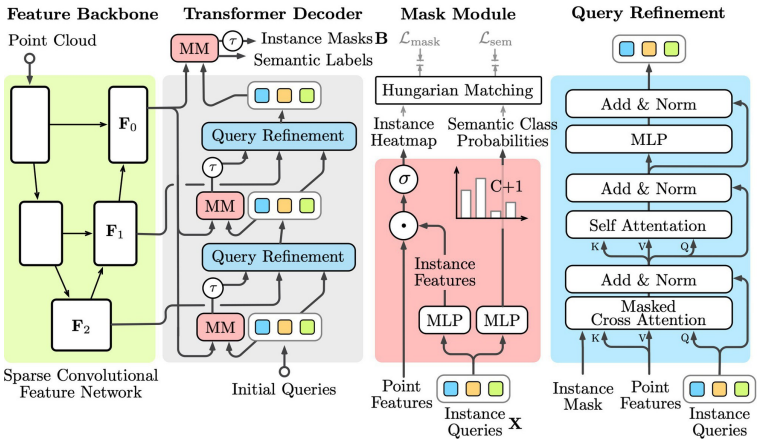


FIGURE 2.31: Mask3D [161] architecture: a 3D instance segmentation model that combines a 3D CNN backbone with a mask prediction head to generate instance masks directly from point clouds.

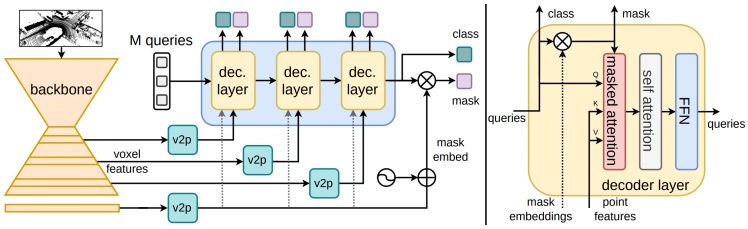


FIGURE 2.32: MaskPLS [122] architecture: a 3D panoptic lidar segmentation model that extends Mask3D to jointly predict semantic and instance labels for point clouds, focusing on outdoor environments.

2.5.1 Proposal-based Methods

Mask3D

Mask3D [161] (see Figure 2.31) is a 3D instance segmentation model that directly predicts instance masks from point clouds. It employs a 3D CNN backbone (Minkowski-UNet) to extract multi-resolution features from the voxelized input point cloud, followed by a mask prediction head that generates instance masks. Instances are obtained through refinement of the so-called queries, or proposals, which are learned embeddings that represent potential object instances. Initally, they are obtained by farthest point sampling(FPS) on the feature map. Thus, if K is the number of queries, K points are sampled and the corresponding features of size D are extracted, resulting in initial queries matrix $\mathbf{X} \in \mathbb{R}^{K \times D}$. For each resolution level r of the backbone, a mask prediction head refines the queries through cross-attentive and self-attentive transformer decoders and predicts instance masks. Finally, instance queries are filtered using the sigmoid function and a fixed threshold of 0.5, to keep the most confident ones. Mask3D achieved state-of-the-art performance on indoor datasets like ScanNetV2 [33] and S3DIS [4], and aerial datasets like STPLS3D [22].

MaskPLS

Mask-based panoptic segmentation in 3D point clouds (MaskPLS) [122] extends the Mask3D architecture to panoptic segmentation in outdoor environments. Similarly to Mask3D, it uses a 3D CNN backbone(MinkowskiU-Net) to extract features from the voxelized point cloud and mask prediction heads to generate instance masks from initial queries (see Figure 2.32), that are progressively refined using decoder-only transformer layers. Differently,

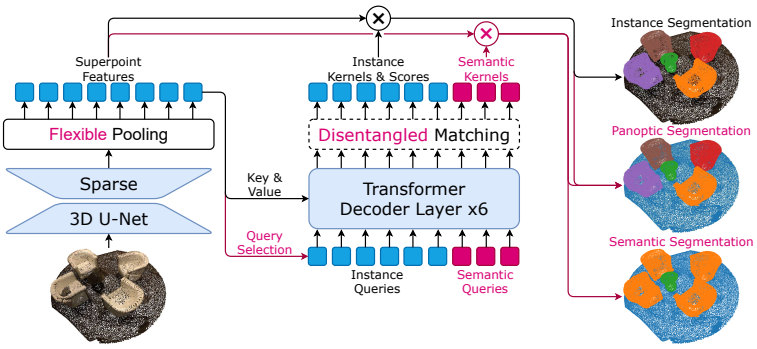


FIGURE 2.33: OneFormer3D [89] architecture: a unified model for 3D semantic, instance, and panoptic segmentation that leverages a transformer-based backbone and task-specific heads.

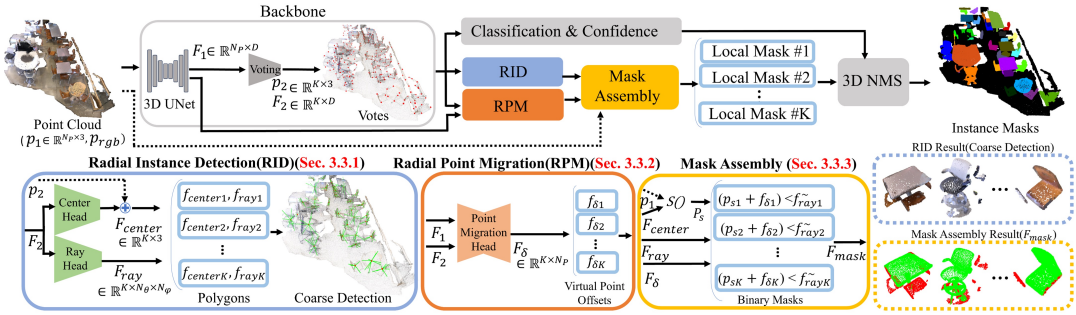


FIGURE 2.34: Spherical Mask [167] architecture: a coarse-to-fine instance segmentation model for colored point clouds that combines 3D CNNs, voting modules, and dynamic convolution for accurate instance mask prediction.

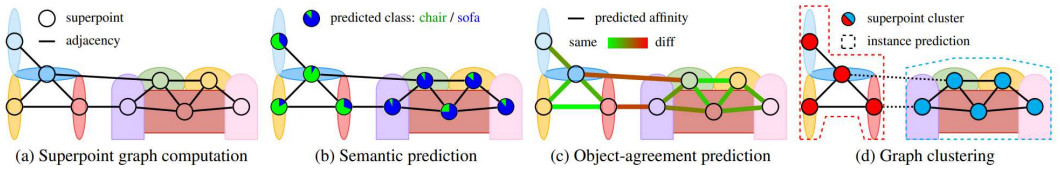
initial queries are not obtained by FPS, but rather are learned embeddings. Also, intermediate supervision is applied at each resolution level of the backbone for both instance and semantic tasks. MaskPLS achieved state-of-the-art performance on the SemanticKITTI [6] and nuScenes [15] datasets.

OneFormer3D

OneFormer3D [89] is a unified decoder-only transformer architecture for 3D semantic, instance, and panoptic segmentation. The model's backbone (see Figure 2.33) is a Minkowski-UNet that extracts multi-resolution features from the voxelized input point cloud. It is followed by a superpoint pooling layer that aggregates point features of each superpoint by averaging them and obtain a single feature per superpoint, as done in SPFormer. These features are then processed by a series of transformer decoder layers that refine the superpoint embeddings. Finally, task-specific heads generate predictions for semantic segmentation, instance segmentation, or panoptic segmentation based on the refined superpoint features. OneFormer3D achieved state-of-the-art performance on indoor datasets like ScanNetV2 [33] and S3DIS [4].

Spherical Mask

Spherical Mask [167] is a coarse-to-fine instance segmentation method for colored point clouds (see Figure 2.34). The model first extracts point-wise features through a 3D CNN backbone (SPConv-based UNet). Then, a voting module predicts K 3D votes, or queries, that represent the 3D location of potential instances. The 3D votes are obtained by FPS, then each spherical neighborhood around the votes is processed by a PointNet++ [148] layer



(a) Superpoint graph computation (b) Semantic prediction (c) Object-agreement prediction (d) Graph clustering

FIGURE 2.35: Superpoint Transformer (SPT) [153] architecture for panoptic segmentation: a lightweight transformer-based model extending SPT for semantic segmentation (Section 2.4.4). Superpoints with compatible class distribution and object agreement are merged to obtain the final panoptic segmentation.

to produce each vote's feature vector. This voting module is applied twice, progressively refining the vote locations. Then, the coarse-part of the model estimates K possible coarse instances by inscribing points into 3D polygons. For each vote, an MLP predicts the instance center and another MLP predicts N 3D rays to build a 3D polygon around the center. It's worth noting that each MLP's input is the 3D UNet processed point cloud features centered at each vote location. Then, the fine-part of the model refines the coarse instances by predicting per-point offsets through dynamic convolution. This module tries to adjust the point locations to better fit the object boundaries. Finally, the classification module, consisting of two MLPs, predicts semantic labels for each point and confidence scores for each instance, followed by Non-Maximum Suppression (NMS) to filter overlapping instances. Spherical Mask achieved state-of-the-art performance on indoor datasets like ScanNetV2 [33], S3DIS [4], and aerial datasets like STPLS3D [22].

2.5.2 Proposal-free Methods

Superpoint Transformer (SPT) for Panoptic Segmentation

Superpoint Transformer (SPT) [154] (see Figure 2.35) is a lightweight (only 212k parameters) transformer-based model for 3D instance segmentation that operates on superpoints. It extends the original architecture (see Section 2.4.4) for panoptic segmentation. This is obtained by computing an object agreement score between two adjacent superpoints, which is computed using another MLP head, indicating the likelihood that they belong to the same object. Graph clustering merges superpoints with compatible class distribution and object agreement, obtaining the final panoptic segmentation. SPT achieved state-of-the-art performance on indoor datasets like ScanNetV2 [33] and S3DIS [4], and outdoor datasets like DALES [189] and KITTI-360 [108].

Superpoint Transformer (SPFormer)

A similar method to SPT is Superpoint Transformer (SPFormer) [176]. SPFormer also operates on superpoints obtained through unsupervised geometric clustering (see Figure 2.36). Differently than SPT, SPFormer first extracts point-wise features through a 3D CNN backbone (Minkowski-UNet), and then aggregates point features of each superpoint by averaging them and obtain a single feature per superpoint (superpoint pooling layer). Then, the network produces instance masks and instance query vectors through two branches. The instance mask branch is an MLP that predicts per-superpoint features. The instance query vector branch is composed, as in MaskPLS, by a series of transformer decoder layers. A set of learnable queries is progressively refined by these layers. Finally, instance and query features are fused and task-specific heads generate predictions for semantic and instance segmentation. SPFormer achieved state-of-the-art performance on indoor datasets like ScanNetV2 [33] and S3DIS [4].

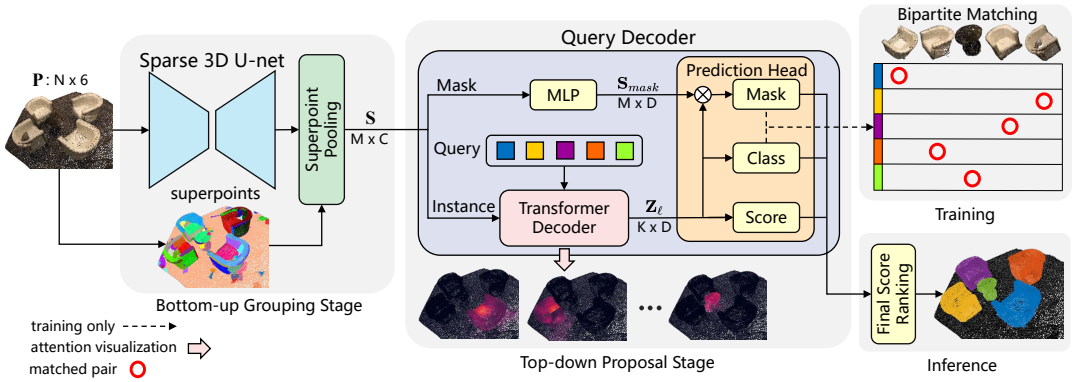


FIGURE 2.36: Superpoint Transformer (SPFormer) [176] architecture for instance segmentation: a transformer-based model operating on superpoints, capturing long-range dependencies through attention mechanisms.

2.6 Traversability analysis

Among the 3D vision tasks, traversability analysis is particularly relevant for mobile robotics, as it directly impacts navigation safety and efficiency. This task encompasses a range of techniques and methodologies aimed at assessing the navigability of terrain for robotic platforms.

Traversability analysis is a special case of (binary) semantic segmentation by which a robot assesses whether regions of the environment can be safely crossed. It plays a critical role in navigation, as poor estimation of terrain difficulty or obstacle presence can lead to unsafe trajectories or mission failure.

Classical approaches rely on terrain modeling and the use of cost maps, where geometric or appearance-based hand-crafted features are converted into spatial representations that guide planning. Obstacles are detected and classified to define non-traversable areas, while traversable regions are assigned varying costs depending on slope, roughness, or other criteria. More recently, learning-based methods have emerged, leveraging large datasets to infer traversability directly from raw sensor data. These methods can capture complex patterns that go beyond hand-crafted rules but may require significant training data and computational resources.

Both hand-crafted features- and learning-based approaches are widely studied, and hybrid methods are increasingly common in modern robotic systems.

2.6.1 Computational considerations

The effectiveness of a navigation pipeline is tightly coupled with its computational feasibility. Many state-of-the-art perception and planning methods are designed to exploit the parallelism of graphics processing units (GPUs), which are highly efficient at performing large numbers of simple operations in parallel. This makes them particularly well-suited for deep learning workloads, dense point cloud processing, and image-based tasks. However, GPUs are often power-hungry, generate significant heat, and require additional hardware space, making them less suitable for embedded or resource-constrained robotic platforms.

In contrast, central processing units (CPUs) are general-purpose processors optimized for sequential task execution and complex logic. While CPUs typically offer fewer cores and less raw parallelism than GPUs, they are more flexible, energy-efficient, and easier to integrate into compact robotic systems. Moreover, they are generally cheaper than GPUs. As a result, many field robots rely primarily on CPU-based computation, especially when operating in environments where low power consumption, robustness, and long-duration autonomy are critical.

For this reason, algorithm design for mobile robotics must carefully consider the trade-offs between CPU and GPU implementations. GPU-based approaches may provide higher accuracy or faster processing for perception tasks but can be impractical for long-term deployments. CPU-oriented solutions, while more constrained in computational capacity, can offer the robustness and efficiency needed for real-world autonomy.

2.7 Sonar-based Underwater Visual Place Recognition (VPR)

Visual Place Recognition (VPR) typically involves matching a query image to a database of reference images and finding the most similar one, usually exploiting the extraction of distinguishing features for comparison. Handcrafted local [117, 75] and global [136] features that typically excel with optical images often fall short in the complex conditions of sonar imaging. To address this, [103] introduced the use of CNN-learned features as descriptors, marking a shift towards deep learning solutions. Two common techniques adopted in deep learning-based VPR can be are categorized as graph-matching, similarity learning and contrastive learning.

2.7.1 Graph-comparison methods

Santos et. al.

Santos et al. [159] proposed a graph-based method for sonar image matching for loop closure detection using similarity learning. First, a query sonar image is pre-processed to remove noise and enhance sonic features. Then, a segmentation module extracts the regions with higher acoustic intensity variations, which are likely to correspond to meaningful structures in the environment. The peaks of intensities higher than a predefined intensity threshold are used to guide the segmentation. Segmented regions are represented as gaussian functions. A graph is constructed where nodes correspond to source and destination gaussians, and edges connects two nodes. An edge is created between two nodes if the Euclidean distance between their centroids is lower than a predefined distance threshold. This graph construction is meant to be produce an image description invariant to translation and rotation. Then a graph-matching algorithm compares the query graph to a database of reference graphs, computing a similarity score for each comparison. The method is evaluated on real sonar datasets, Aracati 2014 [169] and Aracati 2017 [159], showing promising results in loop closure detection.

2.7.2 Similarity Learning

Sonar Matching Network

Sonar Matching Network [14] proposed a siamese CNN architecture to compute similarity scores for sonar images matching. The CNN takes as input a pair of sonar images and processes them through convolutional layers to extract feature representations. A final head computes a similarity score, resembling the intersection value between the two sonar images, using the known relative pose between the two images and the sonar sensor parameters.

2.7.3 Contrastive Learning

PoseNet with Triplet Loss

A notable advancement came from [151], which adapted PoseNet [79], based on GoogLeNet [178], with a triplet loss. Triplet loss is a popular technique in face recognition; it achieves strong

results also on synthetic and real sonar datasets. Essentially, a triplet of sonar images is sampled from the training set during training: an anchor image, a positive image (nearby the place of the anchor), and a negative image (from a different place). The triplet loss encourages the network to learn feature embeddings such that, in the feature space, the anchor and positive images are closer together than the anchor and negative images by a specified margin. The method showed that deep learning could effectively learn robust sonar image descriptors for place recognition even in real world datasets: Aracati 2014 [169] and Aracati 2017 [159].

ResNet-18 with Triplet Loss

Similarly to PoseNet with triplet loss, Larsson et al. [96] leveraged triplet loss with ResNet-18 to capture spatial metrics in side-scan sonar data. Interestingly, to measure the proximity of two sonar images the authors use mIoU between bathymetry meshes. The ResNet-18 is trained for binary classification: given a pair of sonar images, the network predicts whether they are from nearby places or not. The accuracy metric is used as a proxy for place recognition performance. The method is evaluated on a custom, real side-scan sonar dataset, showing promising results in place recognition on different true overlap levels between sonar images.

ResNet-18 with Triplet Loss and Random Gaussian Projection (RGP)

Recently,Donadi et al. [36] proposed a sonar image descriptor-based place recognition system where the descriptor is computed by a ResNet-18 and a random Gaussian projection (RGP) layer. First, sonar images are pre-processed in a three-step pipeline: (i) normalization to uniform insonification, (ii) discrete wavelet transforms (DWT) filtering, and (iii) thresholding with a constant false alarm rate (CFAR), obtaining binary images. The triplet loss is used to train the network to produce discriminative sonar image descriptors of size 128. Descriptors matching for loop closure detection is performed through cosine distance between descriptors. The system is trained only on synthetic datasets, and tested on both synthetic and real datasets (Aracati 2014 [169], Aracati 2017 [159] and Wang2022 [197].). Area under the precision-recall curve (AUC-PR), recall at 95% precision (R@95P) and precision over FOV overlap show that the proposed method performs well also in real sonar datasets, despite being trained only on synthetic data.

Chapter 3

P-SVM: Pyramidal 3D Feature Fusion on Polar Grids for Fast and Robust Traversability Analysis on CPU with SVMs

UTONOMOUS vehicles and ground robots rely on accurate traversability analysis to navigate safely. In this chapter we present P-SVM, our machine learning–based approach that combines geometric features with a pyramid-polar space representation using support vector machine(SVM) classifiers. By fusing fine- and coarse-level spatial information and optimizing implementation details, our method delivers reliable real-time performance even on CPU-only hardware. We compare it against state-of-the-art deep learning approaches on outdoor driving datasets, highlighting its competitive accuracy, faster runtime, and significantly lower energy consumption. We release the open-source implementation to support reproducibility and further research.

3.1 Motivation

Traversability analysis is a core task in the fields of robotics and autonomous driving, providing the basis for safe and effective navigation.

Its complexity depends strongly on both the type of vehicle and the diversity of the terrain [140]. A location is generally considered non-traversable if its characteristics exceed thresholds defined by the vehicle's specifications, such as size, payload, or risk tolerance [99].

A binary classification of terrain into traversable and non-traversable regions must be both reliable and efficient. Reliability is directly tied to vehicle safety, while real-time performance is essential for timely reactions during navigation.

We propose P-SVM, a real-time traversability analysis pyramidal method based on machine learning and polar grid segmentation, using 3D LiDAR point clouds. Unlike many state-of-the-art deep learning approaches, which achieve high accuracy but rely on expensive GPU hardware and often struggle to meet real-time constraints, our method runs efficiently on CPU-only devices. This makes it suitable for low-cost, lightweight platforms without compromising accuracy.

In sum, the chapter presents three key contributions: (i) the development of P-SVM, a geometric-based approach for a support vector machine (SVM)-based traversability analysis and a feature fusion module on a pyramid-polar space representation, that are able to run in

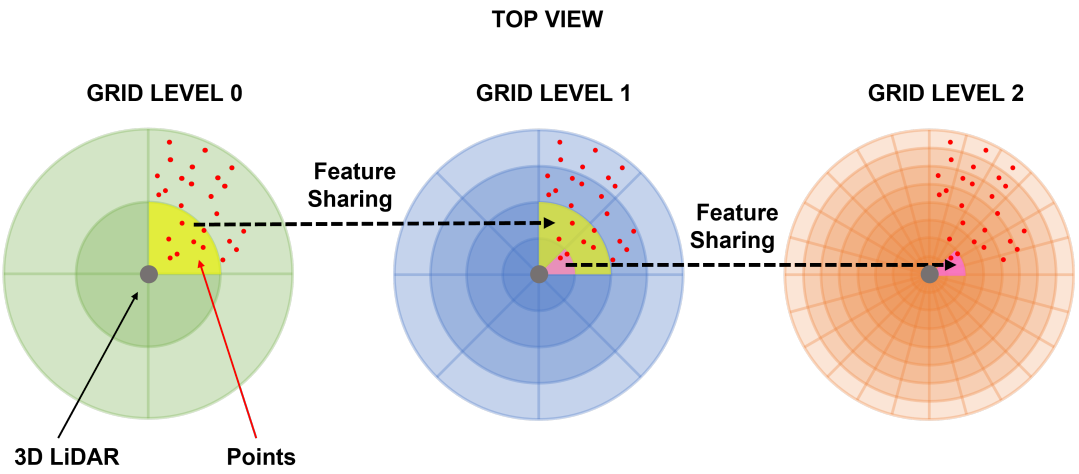


FIGURE 3.1: Pyramid-polar space representation of the input 3D LiDAR point cloud. From left to right, grids progress from coarse (level 0) to fine (level 2). Feature sharing is illustrated: cells at each level inherit features from their corresponding parent cell in the coarser grid.

real-time on CPU-only devices; the comparison of the proposed approach with state-of-the-art semantic segmentation based methods on large publicly available datasets; an open-source implementation of the proposed method is released for public usage [1].

3.2 Related Work

Even though traversability analysis is fundamental in autonomous navigation, there is no explicit formulation of traversability. Usage of this concept can be found in the motion planning context where "traversability maps" are used for the classification of a cell-divided map into traversable or non-traversable categories [100, 171, 82, 188]. Traversability analysis relies on sensor data, such as LiDAR, radar, or cameras, to infer which regions of the environment are safe to traverse. It can be seen as a binary semantic segmentation, where each element (e.g., pixel, voxel) is classified as either traversable or non-traversable. Many semantic segmentation approaches, particularly deep learning-based ones, can therefore be adapted for this task by remapping the label set. For this reason, we also review relevant deep learning-based segmentation methods developed for outdoor environments.

3.2.1 Traditional Approaches

Early work on traversability focused on machine learning with handcrafted geometric features extracted from point clouds. For example, in [20] point clouds are converted into 2D histograms, and a linear classifier distinguishes traversable areas from obstacles. In [209], voxelized point clouds are clustered and classified with random forests, with predictions refined using conditional random fields (CRFs). A semi-supervised approach is presented in [174], where naive Bayes classifiers infer traversability in occupancy grids from limited labeled data. Within the LeGO-LOAM framework [163, 165], Bayesian generalized kernel inference is used to address data sparsity in elevation grid maps for traversability mapping. Finally, in [7], support vector machines combine geometric and appearance-based descriptors, showing that while geometric features alone yield strong performance, integrating visual cues improves robustness and accuracy.

[1] Code available at: https://github.com/Bender97/TraversabilityAnalysis

3.2.2 Deep Learning-Based Approaches

With the rise of deep learning, also traversability analysis moved towards deep architectures, with a strong focus on image-based methods. These methods usually rely on effective and ready-to-use convolutional neural networks (CNNs) designed for multiclass semantic segmentation. We already discussed some of these architectures in Section 2, here we address specific architectures for traversability analysis.

GONet [64] leverages a generative adversarial network (GAN) to predict the traversability directly on RGB images. In [65], the authors address dynamic environments. A CNN requiring the last two acquired images and a virtual navigation command, i.e., a linear and angular velocity, predicts how the static and dynamic elements will propagate in the future frame. A final GONet module predicts the traversability map. OFFSEG [193] performs RGB image semantic segmentation, then regions predicted as traversable are further classified in terms of color clusters to determine sub-classes like mud, puddle, grass, water, etc., useful in off-road scenarios. In [66] the traversability prediction is further divided into categories based on the movement modalities of the robots (e.g. legged, wheeled). In [51] the authors present a semi-supervised learning framework for Mars terrain segmentation, which will be used for studying the traversability of the inspected area. The traversability analysis is initially tackled in an unsupervised setting, with a final supervised training for the traversability classification. [168] proposes a network that learns a function that maps images to scores representing terrain traversability. This function is then incorporated as a term into the path planner's cost function, enabling the consideration of terrain traversability during path planning.

Deep learning architectures are also used to segment 3D point clouds. We've already discussed the main categories in Section 2; here we address specific architectures.

In the projection-based methods lie, for instance, SqueezeSeg [210]. It projects the point cloud onto a sphere and then a point-wise encoder-decoder CNN performs an initial prediction on the points that will be then refined by a CRF. In order to obtain higher accuracy, SqueezeSeg integrates an unsupervised domain adaptation pipeline that trains the network on synthetic data and improves its performance on real data [211] and by exploiting spatially-adaptive convolution, a technique for tailoring convolutional filters to process different parts of the image according to the input data [221]. In [16] a top-view image extracted from the input point cloud is fed into a fully convolutional neural network to distinguish between the traversable and non-traversable classes. RangeNet++ [129] projects the point cloud onto a range image, processes it with a CNN, and then projects the 2D semantic segmentation results into the original point cloud by k-nearest neighbors interpolation. PolarNet [236] uses a bird-eye-view projection of a point cloud into a polar grid, in an effort to alleviate the problem of uneven distribution of the points due to the LiDAR sensor, and a CNN. Nonetheless, SalsaNet [1] and its evolution SalsaNext [30] claim that the type of projection, cartesian or polar, does not bring any type of advantage to the segmentation in their work.

Point-based methods directly operate on the raw point clouds. PointNet [147], a groundbreaking work in 3D perception on unordered point clouds, obtains permutation invariance of the input points by exploiting a learned, approximated symmetric function defined in the point set, computing local features on each input point. RandLA-Net [69] first downsamples the point cloud to efficiently process large amounts of data. Downsampling is paired with a feature aggregation step to avoid losing relevant information using an attention mechanism [190]. KPConv [186] introduces a new convolution operation that works directly on point clouds, eliminating the need for an intermediate representation

Voxel grid-based methods regularize the data in a point cloud in order to capture spatial relations with 3D convolutions. In [184], the input is a voxelized point cloud, but the label predictions on the voxels are then refined to an original per-point level using trilinear

interpolation and a CRF module. Cylinder3D [242], uses a cylindrical grid to partition the space to avoid the LiDAR point cloud density non-uniform distribution. A tailored 3D CNN is used to predict labels for the voxels and a further point-wise refinement module is introduced to improve the coarse voxel predictions. In [24] an encoder-decoder model is used to process a voxelized point cloud with a 3D convolutional network to output per-point labels. It comprises an attentive feature fusion module which is able to aggregate local and global context from the voxel grid and an adaptive feature selection module to better extract global context at different scales. In [46], we proposed a hybrid approach combining geometric and appearance-based features for training deep encoder-decoder architectures to detect the traversability score of a uniform planar grid. The method uses both point clouds and RGB camera images. Point clouds are firstly integrated using odometry estimation, then sorted into the cells of the grid. Geometric features are computed for each cell and then appearance-based features are extracted from RGB by projecting each cell to the image plane. Then, the geometric and appearance feature vector is fed to an encoder-decoder model that outputs the traversability score of each cell.

Cluster-based methods process point clouds by clustering points into cluster sharing similar properties. In [95] points are clustered based on geometric similarities, and then a graph convolutional neural network (GCNN) segmentats the super-points. Points belonging to a cluster will share a common label. In [107] a GCNN is optimized using a structure-aware loss function that exploits the structural information contained in the super-points. In [106] the super-point generation is obtained by an encoder-decoder performing both depth-wise and point-wise graph convolutions.

Hybrid methods fuses different sensors information. In [143, 57] and [235] both images and point clouds are processed by a deep neural network to extract context-rich information and to obtain a more robust traversability analysis. In fact, in [143] this approach has been shown to be effective even in unstructured environments. [35] uses super-points as an intermediate representation to connect 3D and 2D features and super-point-based pooling fuses the two features for joint learning. RPVNet [222] designs a fusion network to merge information coming from point clouds, voxel grids, and range images. The three data representations are processed with parallel networks that share information at multiple levels. Lastly, 2DPASS [224] uses both images and point clouds. During training, two parallel networks are trained, one on the point cloud and one on the related image. Knowledge distillation and feature fusion from the image branch to the point cloud branch are used as a form of regularization. At inference time, the image branch is completely discarded and inference is performed on the point cloud only.

3.3 Our Approach

P-SVM combines handcrafted geometric features, principal component analysis (PCA), and SVMs for traversability analysis on CPU-only devices.

Deep learning-based approaches automatically learn hierarchical features that often outperform handcrafted ones, but they require large annotated datasets, high computational resources (e.g., GPUs), and include models that are difficult to interpret [40]. In contrast, handcrafted features have a smaller footprint, can be tailored to specific tasks, and run efficiently on general-purpose or embedded hardware.

For traversability analysis, we demonstrate that carefully designed geometric descriptors, organized in a hierarchical polar grid representation, can still provide a competitive and practical alternative to deep models, especially in real-time and resource-constrained settings.

We define a feature vector of a set of points $S = \{p_1, p_2, ..., p_n\}$ with $p_i \in \mathbb{R}^3$ as

$$\mathcal{F}(S) = [f_1, f_2, ..., f_k]^T, \tag{3.1}$$

where k is the number of features. SVMs determine an optimal decision boundary by solving a convex optimization problem, ensuring a unique global solution [9, 31]. The kernel trick allows nonlinear separation in higher-dimensional spaces, and in our experiments the radial basis function (RBF) kernel outperformed linear and polynomial alternatives, making it the kernel of choice.

To further improve efficiency and reduce redundancy, we apply PCA to compress the feature space, increasing interpretability but at the same time minimizing information loss [74].

3.3.1 Pyramidal-Polar Representation

We assume that the environment is represented by a 3D LiDAR point cloud, which is partitioned into polar grid cells whose size varies with distance from the sensor. Compared to Cartesian grids, this representation provides a more balanced distribution of points, as distant regions, naturally sparser, are aggregated into larger cells. Points outside a fixed interval $[R_{min}, R_{max}]$ are discarded, where R_{min} depends on vehicle size and and R_{max} on the area of interest. To capture both local detail and broader spatial context, we adopt a pyramidal polar representation: the point cloud is divided into multiple polar grids at increasing resolution, from coarse (level 0) to fine (level $n-1$). We can define relations between cells at different levels: Each fine cell $c_{i,l}$ at level $l > 0$ inherits features from its parent cell $c_{j,l-1}$ at the coarser level. These inherited descriptors may include geometric features or predicted labels, and the process can be applied recursively up the hierarchy. In this way, finer cells integrate both local information and higher-level contextual features, improving the robustness of downstream machine learning models for traversability analysis.

3.3.2 Geometric-Based Feature Extraction

The geometric-based features $\mathcal{F}_g(S) = [f_1, f_2, \ldots, f_r]^T$ of a polar grid cell are computed based on the geometrical properties of the set of 3D points that lie in it, where r is the number of geometrical features. These features are mainly computed using the eigenvalues ($\lambda_1 \geq \lambda_2 \geq \lambda_3$) and eigenvectors of the correlation matrix of the points in S. The eigenvectors indicate the principal directions of variance, with v_3 corresponding to the normal to the plane fitted to the point-set S, which is highly relevant in planar environments such as roads.

Assuming that $\lambda_1 \neq 0$, among others we use some of the features proposed in [7] and reported in 3.1, where:

- $\vec{n} = v_3$ is the normal vector to the plane fitted to S;
- $\hat{z}$ is the versor in the z direction in the LiDAR coordinate system;
- z_i is the z-coordinate of the i-th point in S, with $\bar{z} = \frac{1}{|S|}\sum_{i=1}^{|S|} z_i$;
- σ_i is the i-th singular value of the covariance matrix C of S;
- d^2 is the area of the cell that contains S, with d being the side length.

Additionally, we introduce the zeta difference, a feature capturing the maximum elevation variation along the global scene normal, $\vec{n_p}$. zeta difference is designed to effectively distinguish obstacles from flat, traversable regions. In order to obtain a consistent metric, each point inside a cell is projected to the scene's normal vector $\vec{n_p}$. The projection p_p of a point p is obtained as follows:

$$p_p = \langle p, \vec{n_p} \rangle \vec{n_p} \tag{3.2}$$

Let M and m be the points of a cell whose projections have higher and lower values of the z-coordinate. Then:

$$z_{diff} = |M_z - m_z| \tag{3.3}$$

$$\text{linearity} = \frac{\lambda_1 - \lambda_2}{\lambda_1} \qquad \text{sphericity} = \frac{\lambda_3}{\lambda_1}$$
$$\text{planarity} = \frac{\lambda_2 - \lambda_3}{\lambda_1} \qquad \text{omnivariance} = \sqrt[3]{\lambda_1 \lambda_2 \lambda_2}$$
$$\text{anisotropy} = \frac{\lambda_1 - \lambda_3}{\lambda_1} \qquad \text{eigenentropy} = \sum_{i=1}^{3} \lambda_i \log \lambda_i$$
$$\text{sum of eigenvalues} = \sum_{i=1}^{3} \lambda_i \qquad \text{curvature} = \frac{\lambda_3}{\sum \lambda}$$
$$\text{angle} = \arccos(\vec{n} \cdot \hat{z}) \qquad \text{goodness of fit} = \min(\sigma_i(C))$$
$$\text{roughness} = \frac{1}{|S|} \sum_{i=1}^{|S|} (z_i - \bar{z})^2 \qquad \text{normal vector} = \vec{n}$$
$$\text{inverse cardinality} = \frac{1}{|S|} \qquad \text{surface density} = \frac{|S|}{d^2}$$

TABLE 3.1: Geometric features of a point set S computed by the proposed approach to infer the traversability analysis.

Before being fed into the SVM model, all features f are log-transformed into f_t:

$$f_t = \ln(f + 0.0001) \tag{3.4}$$

and normalized to zero mean and unit variance, to ensure comparability across different scales, improving both robustness and training efficiency.

Points from the LiDAR scan are first assigned to polar grid cells based on their radial distance and yaw angle; points outside the grid bounds are discarded. For a detailed description of this process, refer to Algorithm 1. Cells containing fewer than a threshold τ points are marked as unpredictable, while predictable cells are processed to compute feature vectors, eventually expanded with information from coarser levels. For each predictable cell $c_{i,l}$ at level l, a feature is calculated as described in Algorithm 2.

We then apply PCA to reduce dimensionality, highlight the most relevant geometric features, and improve both model simplicity and runtime efficiency.

3.4 Experimental Evaluation

We implemented P-SVM in C++ and performed all the CPU-only experiments on a Laptop PC equipped with an AMD Ryzen 7 5800H CPU (3.2 GHz), 16GB of RAM, and Ubuntu 20.04 (the internal GPU has not been used). All the deep learning-based approaches were tested on a Desktop PC with an Intel Core i9-10920X CPU (3.50 GHz), 32 GB of RAM, an Nvidia Titan RTX GPU and Ubuntu 18.04. As described in 3.4.8, we also run experiments on a Jetson AGX Xavier with an 8-core Nvidia Carmel CPU, a 512-core Nvidia Volta GPU, 16 GB of RAM, and Ubuntu 18.04.

3.4.1 SemanticKITTI Dataset

To evaluate the performance of P-SVM, we used the SemanticKITTI dataset [6], a publicly available dataset for 3D LiDAR semantic segmentation comprising 11 annotated point cloud sequences and 11 sequences with no publicly available annotation, used for testing. In this work we used the first 11 annotated sequences, referred to as *scenario00-scenario10*, in which a fully sensorized vehicle is driven within an urban context, in low traffic conditions.

Algorithm 1: Bin points in polar grid cells at level l

Input:
$P \subset \mathbb{R}^3$: set of points of the point cloud;
$l \geq 0$: cylinder's level;
R_{min}, R_{max}: minimum and maximum radius ranges of the polar grid at level l;
γ_n: number of yaw steps;
r_n: number of radius steps ;
$r_s = \frac{R_{max} - R_{min}}{r_n}$: radius step range (in meters)
Result: The set of point sets $\{S(i,l)\}$, $\forall$ cell c_i at level $l \geq 0$
for *each cell* c_i *at level* l **do**
  $S(i,l) \leftarrow \varnothing$;
end
for *each point* $p \in P$ **do**
  $\rho \leftarrow \sqrt{p_x^2 + p_y^2 + p_z^2}$;
  if $R_{min} \leq \rho \leq R_{max}$ **then**
    $p_\gamma \leftarrow \arctan(p_x, p_y) + \pi$;
    $row \leftarrow \lfloor \frac{\rho - R_{min}}{r_s} \rfloor$;
    $col \leftarrow (\lfloor \frac{p_\gamma \cdot \gamma_n}{2\pi} \rfloor + \frac{\gamma_n}{2}) \bmod \gamma_n$;
    $idx \leftarrow col \cdot r_n + row$;
    $S(idx,l) \leftarrow S(idx,l) \cup p$;
  else
    ignore point p ;
  end
end

They include dynamic agents (e.g., vehicles and people) and natural elements (e.g., grass, terrain, trees).

3.4.2 Traversability Ground Truth Extraction

Point clouds in the SemanticKITTI dataset are not labeled with the traversable and non-traversable labels. To obtain a traversability ground truth, we pre-processed the point labels and propagated them to the polar grid cells they belong to. We considered as traversable the points having label road, sidewalk, parking, lane marking, or other-ground. All the other labels are considered non-traversable. We determined cell labels considering the LiDAR points that are contained in each cell. Let τ be the minimum number of points for a cell to be considered predictable: in all our experiments we set $\tau = 4$. We labeled as unpredictable all cells containing less than τ points, all the other as predictable. We did so mostly to filter our some outliers contained in the SemanticKITTI, where isolated points are labeled with a wrong label. Then, we labeled as non-traversable the cells containing both road and sidewalk points, to avoid a vehicle driving between road and sidewalk.

These choices imply that:

- $\tau = 4$ is sometimes not sufficient for some geometric properties to completely discriminate cells.
- To compare our approach with semantic segmentation methods, we ahve to map predicted point labels traversability labels (see Section 3.4.6).

Algorithm 2: Compute features for each cell, at each level

Input:
L: number of levels ;
$S(i,l)$: set of points of cell c_i at level l ;
τ: minimum number of points for a cell to be considered predictable;
r: number of geometrical features computed for each cell ;
$\hat{c}_i$: cell at level $l-1$ that contains cell c_i at level l, in other words $c_i \in \hat{c}_i$;
$F_g(c)$: set of geometric features of cell c ;
$F_{in}(c)$: set of features that cell c has inherited from cell $\hat{c}$;
$F_{lab}(c)$: predicted label of cell c (only available at levels $l > 0$) ;
Result: The set of feature sets $\{F(i,l)\}$, $\forall$ cell c_i, level $l \geq 0$
for $l \leftarrow 0$ *to* $L-1$ **do**
  for *each cell* c_i *at level* l **do**
    if $\|S(i,l)\| \geq \tau$ **then**
      compute $F_g(S(i,l)) \leftarrow [f_1, f_2, ..., f_r]$;
      if $l \geq 1$ **then**
        compute $F_{in}(c_i) \leftarrow \{F_{in}(\hat{c}_i), F_g(\hat{c}_i), F_{lab}(\hat{c}_i)\}$;
      else
        $F_{in}(c_i) \leftarrow \varnothing$;
      $F(i,l) \leftarrow \{F_g(S(i,l)), F_{in}(c_i)\}$
    else
      label cell c_i at level l as unpredictable

3.4.3 Training and Test Protocols

We evaluated our method on the SemanticKITTI dataset following the standard protocol: Scenario 08 was used for testing, while the remaining ten scenarios served for training. To configure the SVM models, we performed a grid search for hyperparameter selection.

We designed a pyramid-polar space representation with $n = 3$ levels, determined empirically after testing alternative configurations (see Sec. 3.4.4). We associated each level with a dedicated SVM classifier, whose hyperparameters are reported in Table 3.2. During preprocessing, we partitioned each point cloud into polar cells. For each i-th predictable cell $c_{i,l}$ at level l, we assigned ground-truth labels (Sec. 3.4.2) and computed geometric features $\mathcal{F}_g(S_{i,l})$ from the corresponding point set $S_{i,l}$ (Sec. 3.3.2). At the coarsest level ($l = 0$), we trained SVMs using only geometric features. At finer levels ($l' > 0$), we extended the feature vectors by concatenating local features with information inherited from coarser cells, including predicted labels, geometric descriptors, and derived features $\mathcal{F}_{in}(S_{j,l'-1})$ from the containing cell $c_{j,l'-1}$ (Fig. 3.1). This hierarchical feature sharing allowed us to propagate contextual information across resolutions. The same partitioning and feature extraction pipeline was applied during inference, after which predictions were compared with ground truth to compute the evaluation metrics (Table 3.3).

3.4.4 Approaches for Feature Fusion

We tested several fusion strategies to integrate hierarchical information:

- *Geom:* feature vectors consist solely of the 17 geometric descriptors (Table 3.1) plus the zeta difference (Eq. 3.3), all computed at the current level;

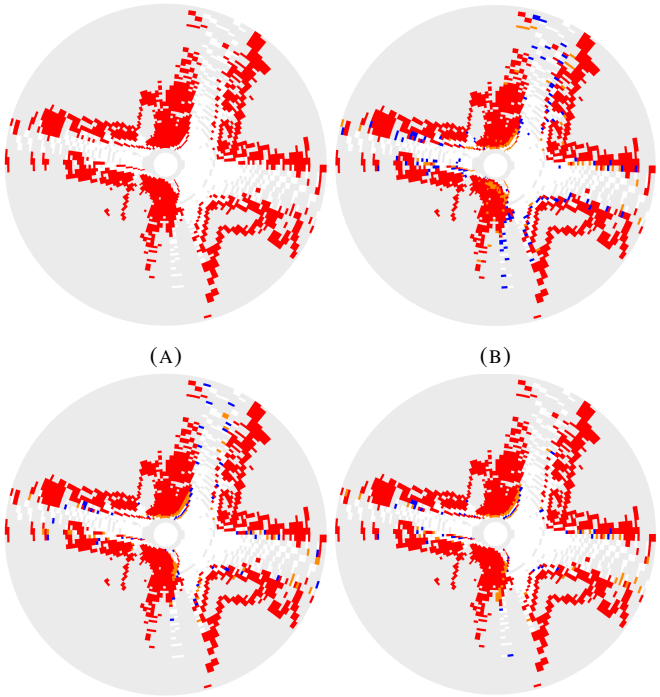


FIGURE 3.2: Polar grid-based traversability analysis applied to a scan of the SemanticKITTI's test Scenario 08: (a) ground truth; (b) our approach; (c) RangeNet++; (d) Point-Voxel KD. Cells correctly classified as traversable and non-traversable are colored in white and red, respectively. Cells incorrectly classified as traversable and non-traversable are colored in orange and blue, respectively. Unknown and unpredictable cells are colored in gray.

- *Geom-label, or Geom-L:* geometric features are augmented with the predicted labels of coarser cells $l-1, l-2, \ldots, 0$, enriching the feature space with broader spatial context. The feature vector has dimension $17+l$;
- *Geom-all, or Geom-A:* geometric features at the current level are augmented with both predicted labels and geometric features from all coarser levels. The resulting dimension is $17 \cdot (l+1) + l$;
- For all strategies, we also applied PCA re-projection to assess the benefit of dimensionality reduction on accuracy and runtime.

3.4.5 Metrics

To compute the metrics we used the number of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). See in depth description in Chapter 2. The performance is measured using the metrics detailed in Table 3.3. Accuracy represents the percentage of input samples that were correctly classified, but it is strongly biased for unbalanced datasets; the intersection over union traversable (IoUT) measures the overlap between the set of ground truth positive samples and the set of predicted positive samples, giving a measure for single-class accuracy; the intersection over union non-traversable (IoUN), measures the overlap between the set of ground truth negative samples and the set of predicted negative samples. When the dataset is dominated by negative samples, the F1 score is a more reliable metric, balancing precision with the rate of detection for the positive class. F1 ignores true negatives and can thus be misleading if the dataset is positive-dominant; the

level	**0**	**1**	**2**
radial steps	8	16	64
yaw steps	16	32	128
ν - nu	0.2028	0.1805	0.1838
γ - gamma	0.098	0.0765	0.1003
C	1.0	1.0	1.0
best method	*geom*	*Geom-L*	*Geom-L*
PCA dimension	17	17	17

TABLE 3.2: Parameters list of the best performing SVM model.

accuracy	$\frac{TP+TN}{TN+FN+TP+FP}$	IoUT	$\frac{TP}{FP+FN+TP}$	TPR	$\frac{TP}{TP+FN}$
F1 score	$\frac{2TP}{2TP+FP+FN}$	IoUN	$\frac{TN}{FP+FN+TN}$	TNR	$\frac{TN}{TN+FP}$
Cohen's kappa	$\frac{2\cdot(TP\cdot TN-FN\cdot FP)}{(TP+FP)\cdot(FP+TN)+(TP+FN)\cdot(FN+TN)}$				

TABLE 3.3: Metrics used for the quantitative evaluation of the models.

Cohen's kappa, which in the binary classification case measures fractional improvement over the random classifier, is symmetric and assesses both directions of predictability. True positive rate (TPR), or sensitivity, measures the rate of detection of TP, conversely true negative rate (TNR) measures the rate of the detection of TN. TNR is particularly important in traversability analysis, where the cost of FP is high as obstacles are not identified as non-traversable cells.

3.4.6 Comparison

We designed our experiments to highlight the contributions of this work, namely the pyramid-polar space representation, the use of geometric features, and the efficiency of SVM-based models enabling low runtime performance.

We compared our framework (P-SVM) against three recent semantic segmentation methods: RangeNet++ (RN++) [129], Cylinder3D (Cyl3D) [242] and Point-Voxel KD (PVKD) [67]. At the time of experimentation, PVKD ranked second on the SemanticKITTI leaderboard and was the top-performing open-source approach not relying on RGB images, followed by Cylinder3D. P-SVM is the only method among those tested that does not use LiDAR remission values yet achieves competitive results.

Quantitative results are reported in Table 3.4, while qualitative examples are shown in Figures 3.2, 3.4, and 3.3. Deep learning methods generally outperform P-SVM, with PVKD leading overall, followed by Cyl3D. Nevertheless, P-SVM achieves competitive performance across all metrics despite its simplicity. It reaches a balanced sensitivity (89.0%) and specificity (93.4%), ensuring reliable classification with consistent F1 scores. IoUN (87.4%) and IoUT (80.4%) confirm that P-SVM effectively handles class imbalance, while Cohen's kappa values indicate substantial agreement with ground truth.

The main distinction lies in runtime efficiency. While PVKD and Cyl3D achieve top accuracy, they require expensive GPUs and still fall short of real-time performance. RN++ is faster but still GPU-dependent. In contrast, P-SVM runs in real time on CPU-only hardware, requiring just 3% of RN++'s runtime and less than 0.2% of PVKD's runtime on the same CPU. This demonstrates that P-SVM provides a strong trade-off: accuracy comparable

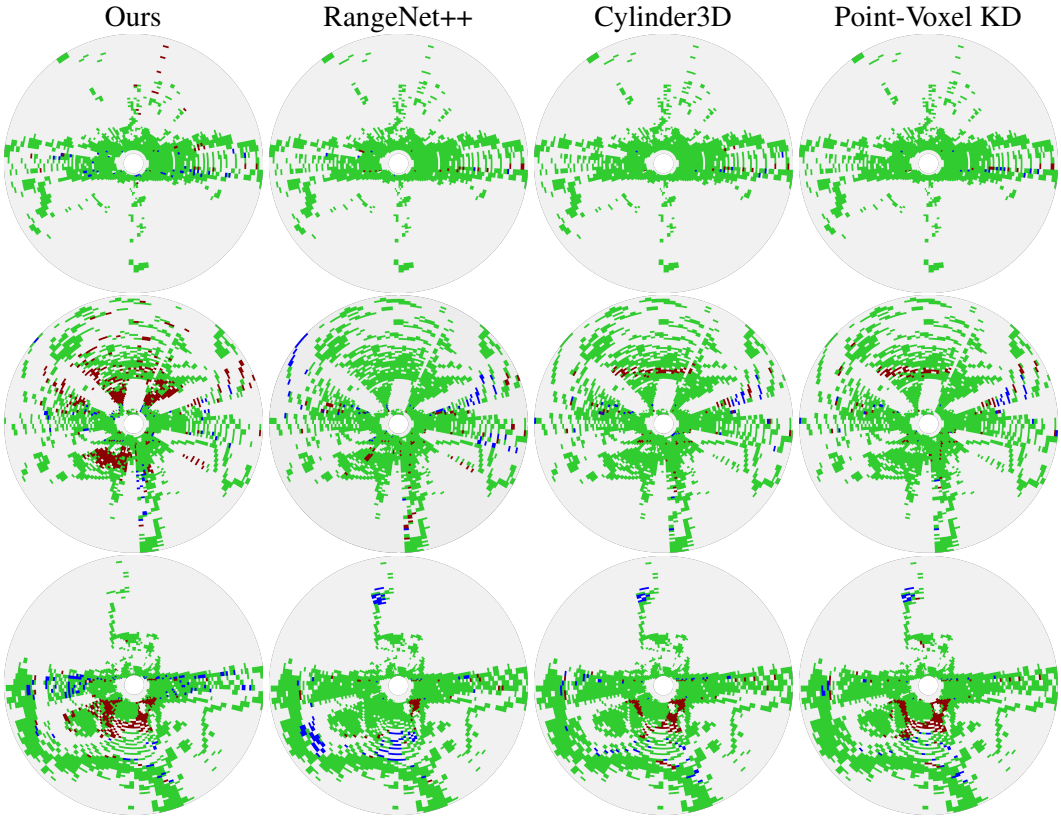


FIGURE 3.3: From left to right, the qualitative results are respectively: our approach, RangeNet++, Cylinder3D, and Point-Voxel KD. Each row in the figure corresponds to different scenes, specifically arranged from top to bottom: a scene where all the methods perform well, the worst-case scenario for our method, and the worst-case scenario for all approaches. Please refer to Fig. 3.4 for the color legend.

to modern deep learning methods while demanding nearly two orders of magnitude fewer computational resources.

3.4.7 Generalization

To evaluate the generalization capability of our approach, we conducted additional experiments on the PandaSet dataset [217], a large-scale collection of urban driving data. We used the annotated point clouds from the main Hesai Pandar64 LiDAR (64 channels, 360° horizontal FOV, 10 Hz), which is comparable to the Velodyne HDL-64E sensor used in SemanticKITTI. Without retraining, we directly tested the models trained on SemanticKITTI on the PandaSet test set, which contains 6080 annotated point clouds.

As shown in Table 3.5 and Figure 3.5, our method generalizes well, performing on par with PVKD and Cyl3D while outperforming RN++, which suffers a marked accuracy drop. This degradation can be explained by the difference in LiDAR scan patterns: PandaSet uses a non-uniform vertical distribution, partially invalidating the standard projection function employed by RN++. In terms of TPR, our method obtains lower results compared to PVKD and Cyl3D. A lower TPR has the consequence of missing some traversable areas, which can be critical in certain applications, where fast path planning is required. However, this is balanced by a high TNR, indicating that non-traversable areas are reliably identified, which is crucial for safety. Our approach yields similar results in terms of the TNR compared to RN++, clearly outperforming both Cyl3D and PVKD. Our method maintains a CPU runtime

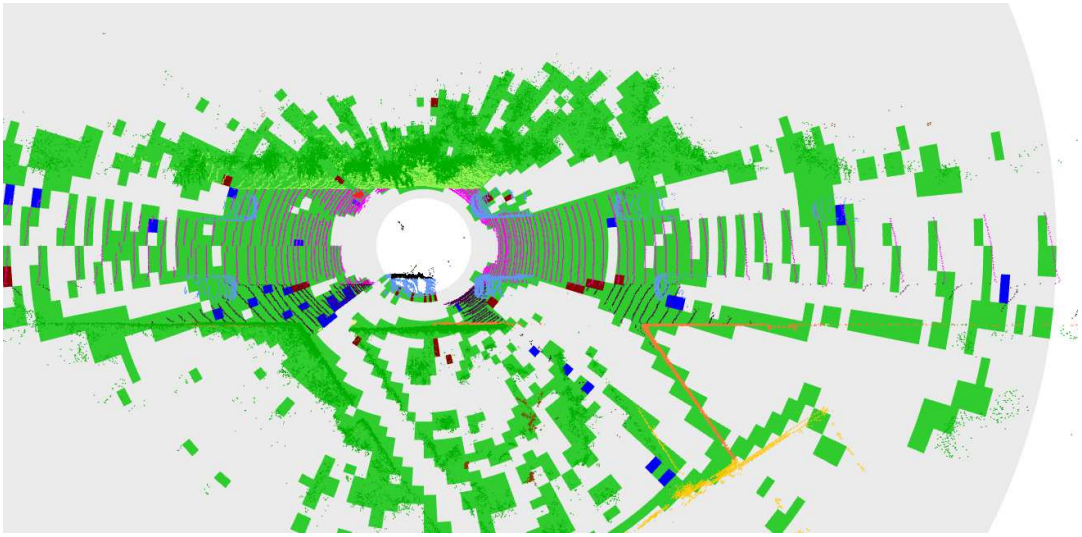


FIGURE 3.4: Qualitative results of the proposed approach applied to a scan from test set Scenario 08. Correctly inferred cells (true positives and true negatives) are colored in green cells, while false positives and false negatives are colored in dark-red and blue, respectively. (For interpretation of the references to color in this figurelegend, the reader is referred to the web version of this article.)

exp	acc %	IoUN %	IoUT %	F1 %	Co %	TPR %	TNR %	Rt_{CPU} ms	Rt_{GPU} ms
P-SVM	91.7	87.4	80.4	89.2	82.4	89.0	93.4	**49**	-
RN++	95.4	92.8	88.6	94.0	90.3	93.2	96.8	1605	**30**
Cyl3D	96.8	95.0	91.9	95.8	93.2	95.3	97.7	6300	110
PVKD	**96.9**	**95.1**	**92.2**	**95.9**	**93.4**	**95.6**	**97.7**	27000	480

TABLE 3.4: Average results on SemanticKITTI Scenario 08.

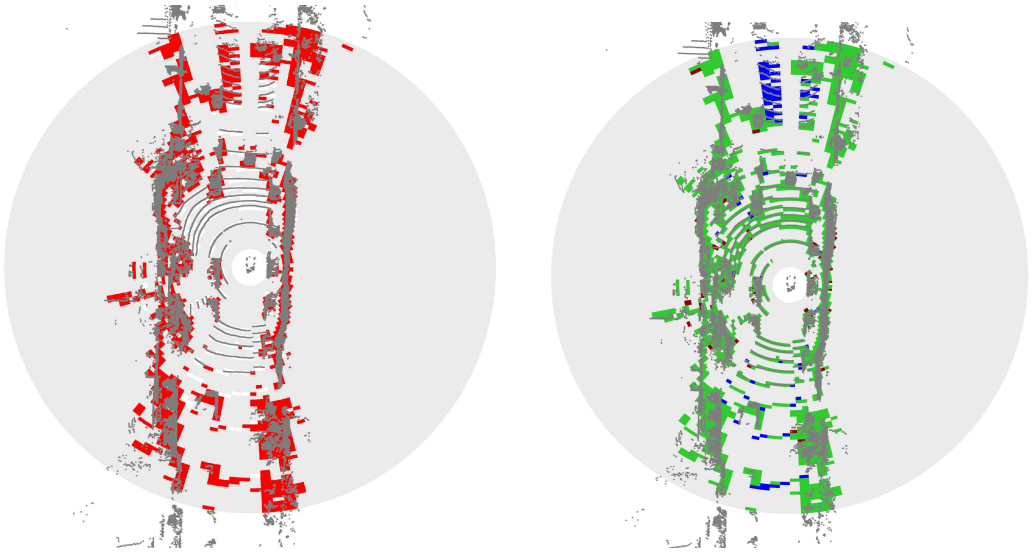
of 50 ms, which remains highly efficient compared to competing methods on both CPU and GPU. Only RN++ on GPU is faster, but still less generalizable.

3.4.8 Performance on Embedded Devices

We also benchmarked runtime and power consumption on a Jetson Xavier embedded board, equipped with a low-power GPU optimized for autonomous navigation tasks. Results (Table 3.6) show that P-SVM, running on CPU, outperforms all deep learning methods in both runtime and energy efficiency. With runtime close to the real-time constraint of 100 ms and more than 70% less power consumption of deep learning-based approaches', P-SVM is well-suited for embedded deployment.

3.4.9 Ablation Study

We further evaluated the different feature fusion variants of our method (*Geom*, *Geom-L*, and *Geom-A*, with and without PCA). At level $l = 0$, the best model applies PCA with dimensionality preserved at 17, achieving the highest accuracy, F1, and IoU while keeping runtime under 4 ms. At level $l = 1$, the *Geom-L* variant with PCA to 17 dimensions performs best, significantly reducing FPs and confirming the benefit of combining label inheritance with PCA projection. Across levels, methods exploiting label inheritance consistently outperform pure-geometric ones, showing that the model leverages both inherited labels and geometric features rather than relying solely on coarse predictions.



(A) (B)

FIGURE 3.5: Qualitative results of the proposed approach in a frame from PandaSet dataset. In the left figure 3.5a, in white are cells classified as traversable, in red as non-traversable, in dark gray *unknown* cells. In the right figure 3.5b, green cells are correctly classified cells (TP and TN), dark-red cells are FP and blue cells are FN. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Exp	Acc %	IoUN %	IoUT %	F1 %	Co %	TPR %	TNR %	Rt_{CPU} ms	Rt_{GPU} ms
P-SVM	81.0	71.2	64.0	78.1	61.5	71.8	89.2	**50**	-
RN++	66.4	58.6	36.1	53.1	30.9	40.3	**89.8**	1800	**35**
Cyl3D	86.2	76.5	75.1	85.8	72.5	87.9	84.7	6700	115
PVKD	**87.9**	**78.6**	**78.4**	**87.9**	**75.9**	**92.6**	83.9	30000	490

TABLE 3.5: Average of the results obtained on PandaSet.

Runtime analysis reveals that feature vectors under 20 dimensions typically run within 30–50 ms, with the best model at 49 ms. In contrast, *Geom-A* variants have runtimes exceeding 200 ms due to their larger feature size. For this reason, in all experiments we adopted the *Geom-L* variant for levels $l > 0$.

3.5 Conclusion

Our P-SVM has proven to be competitive against state-of-the-art deep learning approaches: it gives slightly worse results but with a lower runtime and power consumption.

In relation to navigation risk, our approach yields similar results in terms of the TNR compared to the best deep learning-based method. However, as depicted qualitatively in Figures 3.3 and 3.5, it exhibits minimal drawbacks while maintaining a high level of traversability segmentation quality. Typically, critical regions are accurately classified, and any errors that occur can be corrected through post-processing techniques such as outlier removal filters or filters that leverage successive grid integration. The scalability to multiclass segmentation represents the main limitation of our approach. While our method performs well in segmenting two classes, specifically traversable and non-traversable, it is not suitable for general semantic segmentation tasks with several target classes. In contrast, deep learning

Method	Rt_{CPU} ms	Rt_{GPU} ms	Power Consumption W
P-SVM	**110**	-	**8.260**
RN++	3700	**320**	28.250
Cyl3D	12000	1020	27.570
PVKD	33600	3950	31.700

TABLE 3.6: Runtime and power consumption measurements of the different methods on a Jetson Xavier board.

	Model	PCA	Acc %	IoUN %	IoUT %	F1 %	Co %	TPR %	TNR %
Level 0	*Geom*	-	97.4	97.3	55.6	71.5	70.2	66.7	98.9
	Geom	**17**	**97.5**	**97.4**	**56.1**	**71.9**	**70.6**	66.7	**98.9**
	Geom	15	97.4	97.3	55.5	71.4	70.0	66.7	98.9
	Geom	13	97.4	97.3	56.1	71.9	70.5	**68.8**	98.8
Level 1	*Geom*	-	94.0	93.2	67.5	80.6	77.1	**81.6**	96.3
	Geom-L	-	94.1	93.2	67.3	80.4	76.9	79.7	96.6
	Geom-A	-	93.6	92.7	65.6	79.3	75.5	80.4	95.7
	Geom	17	94.1	93.2	67.3	80.5	77.0	80.3	96.6
	Geom-L	**17**	**94.2**	**93.4**	**67.8**	**80.8**	**77.4**	79.6	**96.8**
	Geom-A	34	93.8	93.0	66.6	80.0	76.3	80.4	96.2
Level 2	*Geom*	-	91.1	86.4	79.5	88.6	81.3	90.1	91.7
	Geom-L	-	91.5	87.2	80.0	88.9	82.1	88.3	**93.5**
	Geom-A	-	90.8	85.9	78.9	88.2	80.6	90.3	91.1
	Geom	17	91.3	86.7	79.7	88.7	81.6	89.6	92.4
	Geom-L	**17**	**91.7**	**87.4**	**80.4**	**89.2**	**82.4**	89.0	93.4
	Geom-A	51	90.6	85.6	78.6	88.0	80.3	**90.6**	90.6

TABLE 3.7: Comparison of the different feature fusion variants. For a description of each method, refer to Section 3.4.4. For each one, we report the best results obtained using grid-search for parameter tuning. The first level, $l = 0$, is the coarsest, while at $l = 2$ we have the finest level, which corresponds to the final traversability level.

methods can be easily adapted to multiclass segmentation by simply adjusting the number of classes in the outer layer, while our method would require a new design process for hand-crafted feature engineering. However, P-SVM runs completely on CPU without the need for high-end GPUs like the other methods. Power consumption measurements demonstrated that it consumes over 70% less power compared to the deep learning approaches. Additionally, performance evaluations done on a Jetson Xavier board have demonstrated that P-SVM is well-suited for embedded devices with limited CPU capabilities, enabling quasi-real-time execution. In contrast, other methods fail to meet real-time constraints even when exploiting the provided embedded GPU.

Acknowledgment

This project has been partially supported by FlexSight Srl company.

Chapter 4

Exploiting Local Features and Range Images for Small Data Real-Time Point Cloud Semantic Segmentation

SEMANTIC segmentation of point clouds is crucial for autonomous driving. While voxel- and point-based methods yield high accuracy, they are computationally expensive; range-based methods are efficient but less expressive. In this chapter, we present our WaffleAndRange method that combines 3D spatial and range image representations, enhanced with a CUDA-based KDTree for fast querying and projection. Experiments on SemanticKITTI and nuScenes show strong performance in both small-data and full-data setups, with a reduced model achieving real-time efficiency for practical deployment. The code of our method is available at https://github.com/Bender97/LocalFeaturesSemSeg.

4.1 Motivation

Point cloud semantic segmentation has recently gained significant attention due to its relevance in autonomous driving, robotics, and environmental perception [58]. Accurate segmentation is crucial for real-time decision-making in autonomous navigation, yet achieving a balance between accuracy and computational efficiency remains challenging. The problem becomes even harder when only limited annotated data are available, where complex state-of-the-art models often fail to generalize and may perform comparably well or even worse than low-complexity neural networks [13].

Recent approaches have made significant advancements in the field by leveraging various LiDAR point cloud representations, including point-based [186, 69, 225], voxel-based [242], pseudo-image [129, 3, 90], and hybrid representation [222, 67, 224]. Notably, hybrid representations generally outperform other classes of methods by taking advantage of the strengths of each representation. However, while the richness of information benefits performance, it introduces drawbacks in terms of processing time, making them unsuitable for real-world scenarios.

WaffleAndRange combines the benefits of both domains. We build on WaffleIron [145], which extracts local geometric features through point embeddings, and RangeFormer [90], which exploits range images for efficient dense processing. By integrating these representations, our network leverages 3D geometry while maintaining fast computation. To further improve runtime, we introduce a GPU-based KDTree for efficient point querying and optimize projection operations with lightweight, element-wise computations that avoid sparse-matrix overhead. This design motivates a framework that can perform real-time segmentation while retaining high accuracy, even in data-limited scenarios.

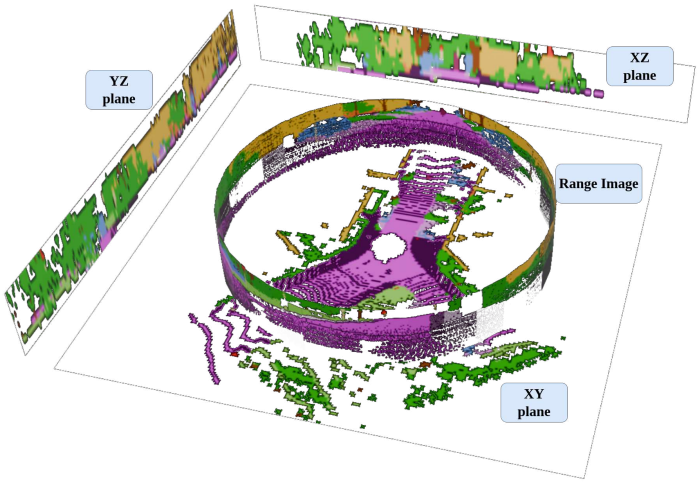


FIGURE 4.1: The four 2D projections utilized by our system for semantically segmenting the 3D point cloud are as follows: XY, XZ, YZ, and range image projection.

The key contributions of this work are: (i) WaffleAndRange, a novel deep learning architecture integrating two state-of-the-art models, designed to be particularly effective in scenarios with limited data availability; (ii) through extensive experiments, we show that the extraction of local features within the embedding module, coupled with the integration of range image data, significantly enhances performance in scenarios characterized by limited data availability; (iii) we drastically reduce the system runtime to just 180 ms per point cloud without any inference optimizer, thanks to code logic optimization.; (iv) we conduct an in-depth performance evaluation on SemanticKITTI [6] and nuScenes [15]; (v) we release an open-source implementation of WaffleAndRange.

4.2 Related Work

In Chapter 2 we provided an overview of the main 3D point cloud semantic segmentation methods. Here, we briefly summarize the main characteristics of each paradigm.

4.2.1 Point-based methods

Point-based semantic segmentation methods directly work on the raw unordered point cloud. PointNet [147] is the pioneer in this field, followed by its improved version PointNet++ [148], using MLPs to obtain permutation invariance of the input points and learn per-point features. KPConv [186], DGCNN [204] and PointConv [213] apply convolution-like operations to exploit local geometric features. RandLA-Net [69] first subsamples the point cloud and then apply an attention mechanism. PointASNL [225] introduces an adaptive sampling module to deal with point clouds with noise. Other recent works [238, 215, 214] rely on the self-attention mechanism introduced in transformers [190] and directly work on the 3D points. However, point-based methods are generally more time-consuming.

4.2.2 Projection-based methods

These methods usually rely on a 2D representation of the point cloud by projecting it onto a surface. In [16] a top-view image is extracted and fed to an fully convolutional network. SqueezeSeg [210] proposes an encoder-decoder network based on SqueezeNet [70] and conditional random fields to refine the predictions. SqueezeSegv2 [211] and SqueezeSegv3 [221]

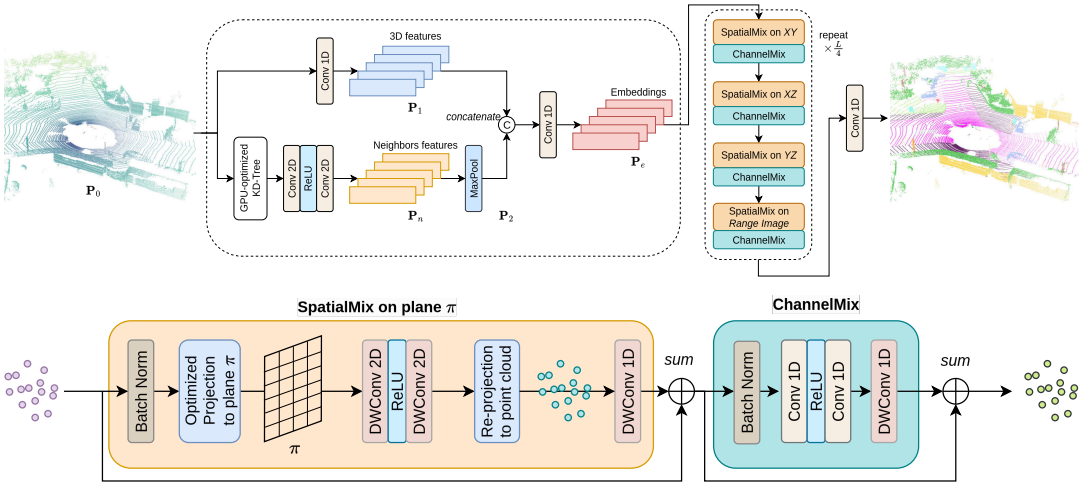


FIGURE 4.2: Top: the architecture of WaffleAndRange, featuring point cloud embedding for point features computation, point cloud processing layers as the backbone with integrated spatial and channel mix modules, and a segmentation head for generating final predictions. Below: details of the backbone, particularly the spatial mix and channel mix modules.

introduce respectively a context aggregation module and a spatially-adaptive convolution to improve the network. RangeNet++ [129] integrates DarkNet as backbone to leverage range images and proposes an efficient k-NN post-processing technique. [236] uses bird's eye view projections. KPRNet [88] exploits KPConv as segmentation head while Lite-HDSeg [150] introduces three different modules and a boundary loss to improve the results. FIDNet [239] and CENet [23] switch encoders to ResNet [63] and substitute the decoders with simple interpolation. Recent works as RangeViT [3] exploits vision transformers (ViTs) [37] pretrained on image data. RangeFormer [90] achieves state-of-the art performance introducing a pyramidal structure based on [201]. WaffleIron [145] proposes a novel 3D backbone, relying on MLPs over 3D points and dense 2D convolution on projection planes. Projection-based approaches usually leverage accelerated computations by operating almost entirely within the 2D image space, thereby attaining real-time efficiency.

4.2.3 Voxel-based methods

These methods first discretize the 3D point cloud into voxel representation and then leverage 3D convolution operators. In [184] the input is voxelized but the label predictions are refined at a per-point level. MinkowskiNet [26] uses sparse convolution [54] to reduce the computational cost. $(AF)^2$-S3Net [24] aggregates multi-scale features with an attention mechanism. Cylinder3D [242] uses a cylindrical grid to partition the space and designs an asymmetrical network to predict labels. Recent studies [93] consider the varying sparsity in LiDAR point clouds, introducing a self-attention mechanism with a radial window. Voxel-based methods leverage the geometric properties of the 3D space, yet they incur higher computational costs when dealing with outdoor LiDAR point clouds. PVKD [67] improves the performance introducing a point-voxel knowledge distillation module while 2DPASS [224] uses RGB images to enrich semantic and structural information during the training process.

4.2.4 Cluster-based methods

These methods first cluster the point cloud into superpoints and then classify each cluster. In [94] the point cloud is over-segmented into geometrically homogeneous clusters and a

graph neural network is used to classify each cluster. In [153] a hierarchical clustering is performed to group points into superpoints and a transformer-based network is used to classify them.

4.2.5 Hybrid methods

Recent trends combine multiple representations (points, projection images and voxels) or RGB images from camera sensors to boost the performance. In [180, 233, 141] fine-grained features at point level are integrated with high-level voxel representations. RPVNet [222] introduces a range-point-voxel fusion network which leverage information from the three distinct representations. Recent works as FRNet [223] combine range image pixels and frustum LiDAR points to boost point-level predictions while UniSeg [112] enriches voxel, range and point-based representations with color and texture information from RGB images. Hybrid approaches leverage the strengths of each representation and may achieve a balanced trade-off between accuracy and efficiency.

4.3 Our Approach

Fig. 4.2 provides a detailed overview of our method. Based on recent state-of-the-art works WaffleIron [145] and RangeFormer [90], it leverages the best of the two paradigms to produce high-quality point cloud segmentation and reduce the runtime of the system. It consists of three main stages: the point cloud embedding (pre-processing), the feature sharing stage, and the final segmentation head. Both WaffleIron and RangeFormer consist of these three stages, at least from a high-level point of view, implementing them in different flavors.

4.3.1 Point Cloud Embedding

Based on theoretical arguments and empirical evidence (see Section 4.4.5), we found that WaffleIron's embedding gives a much more informative representation than RangeFormer's.

WaffleIron's pre-processing operates directly on the 3D point cloud. Firstly, the input point cloud is voxelized and cropped to the sensor FOV, reducing sensor outliers and ignoring points that are too distant. At this point, we obtain a point cloud $\mathbf{P_0} \in \mathbb{R}^{N\times 5}$, where N is the number of points in the point cloud and 5 is the dimensionality of each point's feature, namely x, y, z, LiDAR intensity value, and distance from the sensor (calculated as the Euclidean distance).

P_0 is processed by a simple 1D Convolutional layer that maps the five features to 128, resulting in P_1.

Then, a K-Dimensional Tree (KDTree) is built over the point cloud P_0. We use this data structure to efficiently retrieve the K-Nearest Neighbors (kNN) to each point and build a local feature. Similar to PointNet [147], the local feature extraction captures information such as shape characteristics, local geometry, and point distribution patterns. This hierarchical processing enables the network to learn representations that are robust to variations in scale, orientation, permutation, and density.

The tensor point cloud of neighbors $\mathbf{P_n} \in \mathbb{R}^{K\times N\times 5}$ is obtained by stacking, for each point in $\mathbf{P_0}$, the K neighbors' features. As depicted in Fig. 4.2, $\mathbf{P_n}$, is processed by a series of batch norm, 2D convolutions, and rectified linear unit (ReLU) activation functions. A max pooling operation is applied over the K neighbors to select only the maximum response to preserve the most salient features while suppressing noise, enhancing the representation's robustness to noisy input, and contributing to the permutation and rotation invariance of the network. We denote the output of this processing $\mathbf{P_2} \in \mathbb{R}^{N\times 128}$, having the same number of points and feature size of P_1, thanks to the pooling operation.

Finally, the two point clouds P_1 and P_2 are concatenated and remapped to a feature size of 128 using a 1D convolution, obtaining $\mathbf{P_e} \in \mathbb{R}^{N\times 128}$.

4.3.2 Leveraging 2D and 1D Convolutions

We process the embedding $\mathbf{P_e}$ through a sequence of spatial mix and channel mix operations. These modules act on 2D projections of the point cloud and use only 1D or 2D convolutions, deliberately avoiding 3D convolutions to keep the model lightweight and reduce overfitting (see Section 4.4.5). In this way, 3D data processing is reduced to efficient 2D image analysis.

2D convolutions are well-suited for semantic segmentation as they capture local spatial patterns (edges, textures, shapes) through shared kernels, ensuring parameter efficiency and translation invariance. In contrast, 1D convolutions, though less common in image tasks, are useful here for sequential feature mixing, complementing the 2D operations.

We employ four projections of the point cloud (Fig. 4.1): three orthogonal planes (XY, XZ, YZ) as in WaffleIron [145], and the range image as in RangeFormer [90]. Each projection forms a 2D grid where cell features are computed by averaging all points within the cell, reducing information loss compared to using only the closest point.

A stack of L identical modules parses these projections. Each module:

- projects points onto one of the four 2D planes;
- applies the spatial mix (two depth-wise 2D convolutions + ReLU + grouped 1D convolution);
- inflates the features back onto 3D points;
- applies the channel mix (mirroring the spatial mix but with 1D convolutions).

Both modules include batch normalization and residual skip connections (Fig. 4.2).

4.3.3 Segmentation Head

To fully exploit the local features collected in the preprocessing step, we merge the parsed point cloud with the kNN embedding by simply summing them up. This skip-connection offers evident benefits, such as enhancing the contextualization of point segmentation and facilitating gradient updates to the embedding module during the backpropagation phase.

The final labels are inferred by applying a 1D convolution over the merged point cloud followed by a Softmax layer.

4.4 Experimental Evaluation

4.4.1 Implementation Details

We implemented our method in Python, C++ and CUDA. All the inference experiments were performed on a consumer Laptop PC equipped with an AMD Ryzen 7 5800H CPU (3.2 GHz), 16GB of RAM, and Ubuntu 20.04 (the internal GPU has not been used). All the deep learning trainings were done on a Desktop PC with an Intel Core i9-10920X CPU (3.50 GHz), 32 GB of RAM, an Nvidia Titan RTX GPU, and Ubuntu 18.04. All the results reported refer to the performance obtained at the 45th epoch of training process.

method	**mIoU %**	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
RangeFormer [90]	15.6	47.9	✗	✗	✗	0.2	0.1	✗	✗	62.5	0.1	27.9	1.3	26.6	1.8	57.4	2.6	41.1	15.6	11.8
FRNet [223]	22.0	71.3	✗	✗	✗	0.6	0.1	✗	✗	65.2	0.2	47.4	1.9	52.3	2.0	76.8	2.2	59.5	14.5	24.3
Cylinder3D [242]	23.7	68.7	✗	✗	✗	0.8	1.7	✗	✗	64.0	0.3	44.3	1.1	57.5	8.5	78.8	16.4	57.9	24.6	25.3
WaffleIron [145]	24.6	86.8	✗	✗	✗	0.1	1.5	✗	✗	60.8	0.9	43.6	0.5	55.1	4.8	82.0	18.7	67.6	25.5	19.2
WaffleAndRange	**24.9**	86.4	✗	✗	✗	0.2	1.2	✗	✗	61.9	0.5	47.3	0.5	55.9	4.7	80.4	18.5	66.9	24.9	23.4

TABLE 4.1: Semantic segmentation performance on SemanticKITTI validation set (sequence 8) of methods trained only on Sequence 04. The ✗ symbol indicates classes not present in the training set.

method	**Small data mIoU %**	**Full data mIoU %**
RangeFormer [90]	15.9	67.9
FRNet [223]	22.5	68.7
Cylinder3D [242]	23.2	64.3
WaffleIron [145]	24.6	68.0
WaffleAndRange	**24.9**	**69.0**

TABLE 4.2: Semantic segmentation performance on SemanticKITTI validation set (sequence 8) of methods trained only on Sequence 04 (second column) or full training set (except sequence 8, third column).

4.4.2 Inference time reduction

As pictured in Figure 4.3, we progressively reduced the runtime of our system from an initial 360 ms, to 180 ms, to a final, real-time 80 ms. We obtained this by applying three modifications to the original implementation of WaffleAndRange.

KDTree on GPU: as outlined in Section 4.3, the pre-processing module requires building a KDTree over the input point cloud. WaffleIron originally relied on the SciPy KDTree implementation [192], which is efficient on CPU but not suited for real-time applications. To address this, we adopted cudaKDTree[1], an open-source CUDA/C++ library, and ported the computation to the GPU. This reduced the total time for building and querying the tree from 160 ms to 15 ms.

Flattening operation: in WaffleIron, the SpatialMix module averages point features within the same cell using batch matrix multiplication. This involves constructing a large sparse matrix in $\mathbb{R}^{B,HW,N}$ and multiplying it by a weight matrix in $\mathbb{R}^{N,B}$, where B is the batch size, HW the flattened grid size, and N the number of points. While correct, this approach is memory-intensive and slow, requiring 130 ms across all $L = 48$ layers.

Reducing L: in the ablation study, we discovered that reducing the number of layers from $L = 48$ to $L = 12$ drastically reduces the runtime to a total of 80 ms, making it real-time ($<$100 ms) at the cost of a slight drop in mIoU performance (from 69.0 % to 65.8 %).

We optimized this step by reordering the operations: point features are first element-wise multiplied by the weights, then aggregated using the efficient `scatter_add_` function from PyTorch [142]. This avoids constructing the sparse matrix, reducing the time to 90 ms and saving 40 ms overall. Combined with the GPU KDTree, the total runtime decreased from 365 ms to 180 ms. With only $L = 12$ layers, the model achieves competitive accuracy in 80 ms, enabling real-time inference.

[1] https://github.com/ingowald/cudaKDTree

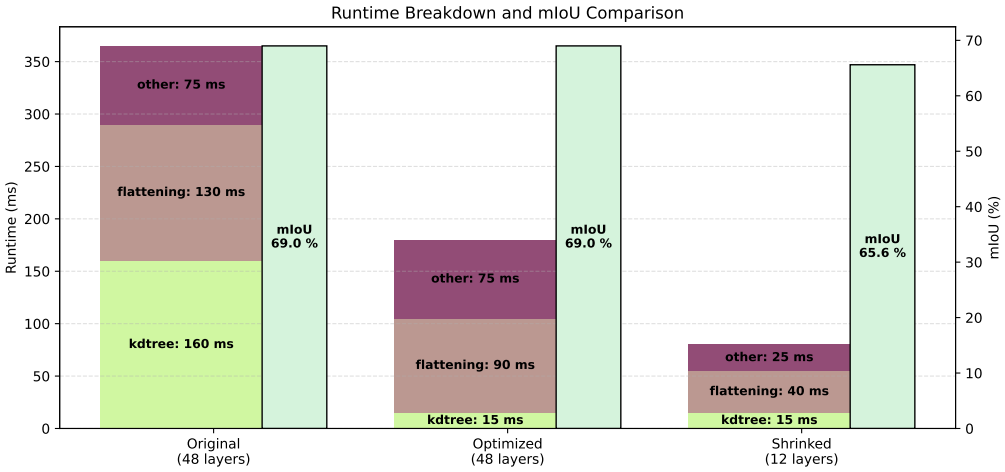


FIGURE 4.3: Runtime breakdown of the three main time-consuming modules of our method: KDTree building and querying (), flattening operation () and all the other operations (). We also plot the mIoU performance (), which slightly drops in order to allow our method to be real-time.

method	**mIoU %**	car	bicycle	motorcycle	truck	other-vehicle	person	bicyclist	motorcyclist	road	parking	sidewalk	other-ground	building	fence	vegetation	trunk	terrain	pole	traffic-sign
SqSegV2 [211]	39.6	82.7	21.0	22.6	14.5	15.9	20.2	24.3	2.9	88.5	42.4	65.5	18.7	73.8	41.0	68.5	36.9	58.9	12.9	41.0
RangeNet++ [129]	52.2	91.4	25.7	34.4	25.7	23.0	38.3	38.8	4.8	91.8	65.0	75.2	27.8	87.4	58.6	80.5	55.1	64.6	47.9	55.9
SqSegV3 [221]	55.9	92.5	38.7	36.5	29.6	33.0	45.6	46.2	20.1	91.7	63.4	74.8	26.4	89.0	59.4	82.0	58.7	65.4	49.6	58.9
CENet [23]	64.7	91.9	58.6	50.3	40.6	42.3	68.9	65.9	43.5	90.3	60.9	75.1	31.5	91.0	66.2	84.5	69.7	70.0	61.5	67.6
2DPASS* [224]	67.4	96.3	51.1	55.8	54.9	51.6	76.8	79.8	30.3	89.8	62.1	73.8	33.5	91.9	68.7	86.5	72.3	71.3	63.7	70.2
WaffleIron [145]	70.8	97.2	70.0	69.8	40.4	59.6	77.1	75.5	41.5	90.6	70.4	76.4	38.9	93.5	72.3	86.7	75.7	71.7	66.2	71.9
RangeFormer [90]	**73.3**	96.7	69.4	73.7	59.9	66.2	78.1	75.9	58.1	92.4	73.0	78.8	42.4	92.3	70.1	86.6	73.3	72.8	66.4	66.6
WaffleAndRange	69.0	96.9	66.6	66.8	30.0	57.6	75.2	74.8	44.6	91.2	68.5	77.1	36.6	92.1	68.0	85.7	74.2	70.3	63.7	70.7

TABLE 4.3: Semantic segmentation performance on SemanticKITTI test set. Regarding 2DPASS*, we report the results of the baseline of [224] trained with lidar data but no images, i.e., in the same setting as the other methods in this table.

4.4.3 Datasets and Metrics

We evaluate on SemanticKITTI [6] and nuScenes [15].

SemanticKITTI comprises 22 sequences, each segmented into 19 classes. Following the standard split, sequences 00–10 (except 08) are used for training, sequence 08 for validation, and 11–21 for testing.

Instead, nuScenes comprises 1000 scenes labeled into 16 classes. Following the official split, we use 700 for training, 150 for validation, and 150 for testing.

We report the intersection over union (IoU) for class i and the average score (mIoU) over all classes.

4.4.4 Performance on Autonomous Driving Datasets

We consider two setups:

- Small data, to stress-test generalization. Models are trained only on SemanticKITTI sequence 04 (271 scans) and validated on sequence 08 (4071 scans). Augmentations include random XY flips, rotations, and scaling.

method	**Small data mIoU %**	**Full data mIoU %**
Rangeformer [90]	15.9	67.9
FRNet [223]	22.5	68.7
Cylinder3D [242]	23.2	64.3
WaffleIron [145]	24.6	68.0
WaffleAndRange	**24.9**	**69.0**

TABLE 4.4: Semantic segmentation performance on SemanticKITTI validation set (sequence 8) of methods trained only on Sequence 04 (second column) or full training set (except sequence 8, third column).

method	**mIoU %**	barrier	bicycle	bus	car	const. veh.	motorcycle	pedestrian	traffic cone	trailer	truck	driv. surf.	other flat	sidewalk	terrain	manmade	vegetation
RangeNet++ [129]	65.5	66.0	21.3	77.2	80.9	30.2	66.8	69.6	52.1	54.2	72.3	94.1	66.6	63.5	70.1	83.1	79.8
PolarNet [236]	71.0	74.7	28.2	85.3	90.9	35.1	77.5	71.3	58.8	57.4	76.1	96.5	71.1	74.7	74.0	87.3	85.7
SalsaNext [30]	72.2	74.8	34.1	85.9	88.4	42.2	72.4	72.2	63.1	61.3	76.5	96.0	70.8	71.2	71.5	86.7	84.4
Cylinder3D [242]	76.1	76.4	40.3	91.2	93.8	51.3	78.0	78.9	64.9	62.1	84.4	96.8	71.6	76.4	75.4	90.5	87.4
2DPASS* [224]	76.2	75.3	43.5	95.3	91.2	54.5	78.9	72.8	62.1	70.0	83.2	96.3	73.2	74.2	74.9	88.1	85.9
WaffleIron [145]	77.6	78.7	51.3	93.6	88.2	47.2	86.5	81.7	68.9	69.3	83.1	96.9	74.3	75.6	74.2	87.2	85.2
WaffleAndRange	**77.6**	78.5	49.6	91.8	87.6	52.7	86.7	82.2	70.1	67.2	79.7	97.0	74.7	76.8	74.9	87.5	85.0

TABLE 4.5: Semantic segmentation performance on nuScenes validation set. Regarding 2DPASS*, we report the results of the baseline of [224] trained with lidar data but no images, i.e., in the same setting as the other methods in this table.

- Full data, standard training with all data. On SemanticKITTI we also apply InstanceCutMix and PolarMix, while nuScenes uses its full training set.

Training was done with $L = 48$ layers, 256-dim features and 40 cm grid for SemanticKITTI, and $F = 384$ features with 60 cm grid for nuScenes.

The results of the small data setup are reported in Table 4.1. Our method outperforms all state-of-the-art baselines, including WaffleIron, improving mIoU by 0.3%. The addition of range image projections improves feature sharing across the Waffle process, boosting generalization.

Our method is also the best performing in the validation sequence of the full data setup (see Table 4.4), in which it obtains a mIoU of 69.0%. This confirms the generalization capabilities that the model obtains in the small data setup. In Section 4.4.5, we conduct a more in-depth analysis of the various components that led us to the architecture development.

The table clearly indicate the superiority of both the original WaffleIron and our method, WaffleAndRange, which obtains slightly better performances on average. The benefit is not as large as in SemanticKITTI probably because the range image is here applied to point clouds obtained from a LiDAR having 32 channels instead of 64 channels as in SemanticKITTI. The range image hence is less informative and the objects are less distinguishable. It's interesting to note that important semantic objects like motorcycle, pedestrian and traffic cones are better segmented by our method. Also, we reach the top results in construction vehicle, drivable surface, sidewalk, terrain and manmade classes. As a side note, to this end, to drastically improve the results of segmentators on average one should focus on improving the results on the vulnerable objects: motorcycle, motorcyclist, bicycle, bicyclist, person/pedestrian, traffic cones, as well as truck, other-ground. Methods perform poorly on these classes mainly for two reasons: (i) they are under-represented classes and (ii) they are

method	range	L	drop	skip	aug	**mIoU %**
WaffleIron [145]	✗	48	-	-	✗	24.4
WaffleAndRange	✓	48	✗	✗	✗	**24.9**
WaffleAndRange	✓	48	✓	✓	✗	22.3
WaffleAndRange	✓	48	✗	✓	✗	23.8
WaffleIron [145]	✗	48	-	-	✓	29.3
WaffleAndRange	✓	48	✗	✗	✓	32.7
WaffleAndRange	✓	48	✗	✓	✓	**32.7**
WaffleIron [145]	✗	12	-	-	✓	28.4
WaffleAndRange	✓	12	✓	✓	✓	20.6
WaffleAndRange	✓	12	✗	✗	✓	28.5
WaffleAndRange	✓	12	✗	✓	✓	**29.8**

TABLE 4.6: Small data setup. Results on methods trained only on SemanticKITTI's sequence 04 (271 point clouds) and tested on sequence 08 (4071 point clouds).

method	range	L	drop	skip	aug	**mIoU %**
WaffleIron [145]	✗	12	-	-	✓	62.6
WaffleAndRange	✓	12	✗	✓	✓	**65.8**

TABLE 4.7: Results on methods trained on full training set of SemanticKITTI and tested on sequence 08 (validation set). Network configuration with only 12 layers and active dataset augmentation.

easily mistaken with other similar classes, such as confusing bicycle with motorcycle, motorcyclist with bicyclist, other-ground with terrain, truck with car, and so on. Methods to mitigate both these challenges include data augmentation techniques, which artificially augment the number of these classes by cutting and pasting instances or crops around a scene, both on the point cloud level or on the projection (e.g., range image) level, or tricks on the loss function which weights more errors on the underrepresented classes, by assigning weights (e.g., in the cross entropy loss) or with Lovász [8] or Dice [173] losses.

4.4.5 Ablation Study

We conducted an ablation study on the SemanticKITTI dataset to better understand the contribution of different design choices. In the small data setup, we tested three techniques: random dropout of neighbor points in the embedding module, skip-connections between neighbor embeddings and backbone outputs, and the effectiveness of data augmentations (InstanceCutMix and PolarMix). Results showed that dropout had little effect, while skip-connections only became beneficial when combined with stronger data augmentations, reaching 32.7% mIoU.

We also evaluated a lightweight configuration with $L = 12$ layers (Table 4.7). The trends remained consistent: dropout decreased performance, while skip-connections offered improvements. Using this reduced setup on the full dataset, our method achieved 65.8% mIoU on sequence 08, outperforming WaffleIron (62.6%) despite using fewer layers. Crucially, this configuration also enabled real-time operation with an average runtime of 80 ms.

4.5 Conclusion

This chapter addressed the challenge of real-time semantic segmentation of point clouds under both small and large data regimes. By combining local feature extraction with hybrid projections of the 3D point cloud and range image, WaffleAndRange, the proposed architecture, improved both accuracy and efficiency. Experiments on SemanticKITTI and nuScenes confirmed that WaffleAndRange generalizes well: it performs strongly with limited data while also remaining competitive in data-rich scenarios. Finally, through GPU-based optimizations and streamlined operations, we reduced runtime significantly, yielding a practical solution for autonomous robotics applications.

Chapter 5

Improving Existing 2D Image Segmentators Performance with Zero-Shot Segmentators

OMPLEMENTARY to LiDAR, vision-based perception remains an essential tool in robotics. However, the performance of traditional 2D image semantic segmentation networks can degrade when transferred across domains. Recently, large-scale zero-shot segmentation models have emerged, offering strong generalization without task-specific training. Zero-shot training refers to the ability of a model to correctly make predictions on classes that were not seen during training. This chapter investigates the use of the Segment Anything Model (SAM) [86] to enhance the performance of existing 2D segmentation methods. We address the need for generalizing vision systems that remain reliable in unseen environments without expensive re-annotation or re-training. SAM is a promptable, open-source zero-shot segmentator, capable of generalization to previously unseen objects and images, eliminating the need for additional training. We explore a strategy in which SAM is guided by checkpoints derived from the masks produced by mainstream segmentators, such as DeepLabv3+ and PVTv2, and the resulting outputs are merged with the original segmentation masks. We extensively evaluate our approach on seven heterogeneous public datasets, achieving state-of-the-art results on two. These experiments demonstrate the potential of integrating SAM into existing segmentation pipelines, and the associated open-source implementation is made publicly available.

5.1 Motivation

Semantic segmentation assigns each pixel in an image to a specific class, enabling fine-grained scene understanding. Deep learning models such as DeepLabv3+ [21] have advanced the field by capturing detailed object boundaries efficiently.

However, these mainstream models often struggle to generalize to unfamiliar objects and novel images, limiting their applicability in real-world scenarios. Promptable zero-shot segmentators like SAM [86] and SEEM [244] overcome this limitation by generalizing to unseen objects without additional training. We propose leveraging SAM and SEEM alongside existing segmentators such as DeepLabv3+ to improve segmentation accuracy on both known and unconventional objects. Our approach uses checkpoints extracted from segmentation masks as prompts for the zero-shot models, then fuses their outputs with the original masks.

Zero-shot segmentators provide two key advantages: they are trained on billions of images, enabling them to outperform conventional models in certain cases, and their architectural and dataset diversity introduces complementary information, which is ideal for ensembles. This fusion strategy consistently improves segmentation quality and generalization,

often achieving state-of-the-art results without retraining.

The main contributions of this work are: (i) we validate our approach on seven heterogeneous benchmark datasets, demonstrating that combining SAM with specialized models enhances performance; (ii) our experiments demonstrate how the fusion of a zero-shot segmentator with a specialized model can rival and even surpass the fusion of two state-of-the-art specialized models, avoiding the expensive retraining; (iii) the open-source implementation of our method is provided to facilitate adoption and further experimentation at https://github.com/LorisNanni/Improving-existing-segmentators-performance-with-zero-shot-segmentators.

5.2 Related Work

Deep learning-based methods have significantly advanced image semantic segmentation, enabling precise pixel-level object classification. In Chapter 2, we discussed the fundamentals of semantic segmentation, including key architectures and techniques. Here, we provide a brief overview of relevant literature.

Early CNN-based architectures like U-Net [157] and SegNet [5] combine encoder-decoder structures with skip connections to capture context and spatial details efficiently.
DeepLabv3+ [21] further improves boundary refinement and multi-scale context through Atrous Spatial Pyramid Pooling, while HRNet [196] maintains high-resolution representations throughout the network for accurate fine-grained segmentation.

Vision Transformers have recently been applied to segmentation, capturing long-range dependencies and contextual relationships. Models such as OCR [230], SegFormer [219], and Pyramid Vision Transformer v2 (PVTv2) [202] combine multi-scale representations with self-attention mechanisms to improve segmentation performance. Additional works enhance segmentation by leveraging semantic or statistical relationships across images [200, 105].

Zero-shot learning addresses the generalization limitations of conventional models when encountering unfamiliar objects. SAM [86] and HQ-SAM [78] demonstrate strong zero-shot performance. They are innovative approach to promptable image segmentation. Trained on a vast dataset comprising more than one billion segmentation masks, SAM exhibits impressive zero-shot generalization capabilities, producing high-quality outputs with minimal prompting. Methods like SAMAug [237] and autoSAM [162] exploit SAM features to boost CNN and transformer-based segmentation, even in specialized domains such as medical imaging.

Combining continuous outputs from multiple models is an effective strategy to improve segmentation performance. Weighted aggregation, averaging, or more complex fusion rules have been applied in both image and multi-modal contexts [92, 87, 132, 127], showing consistent gains over single-model predictions. Such fusion strategies motivate the integration of zero-shot segmentators like SAM with conventional models to enhance accuracy and robustness.

5.3 Our Approach

Motivated by the challenges of the datasets used and the capabilities of the zero-shot semantic segmentation methods SAM and SEEM, we study how these methods can improve the performance of mainstream segmentation approaches on such datasets.

The overall system (Figure 5.1) takes as input an image and a corresponding segmentation mask (source image mask). This mask, either ground truth or model-generated, e.g. by a mainstream segmentator such as DeepLabV3 or point transformer v2, separates the foreground, i.e. the class of interest, from the background. Occasionally, source masks may consist of multiple disconnected regions (“blobs”). From this mask, we extract checkpoints

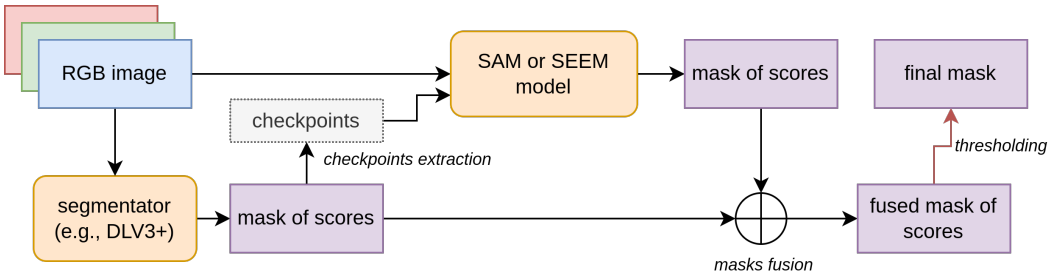


FIGURE 5.1: Visual scheme of the architecture of our system. An input image is processed by a segmentator, e.g., DeepLabv3+. It predicts a segmentation mask of scores. From this mask we extract checkpoints. These are the input, with the original image, to a zero-shot segmentator, e.g. SAM or SEEM. This predicts another segmentation mask of scores. We fuse the two segmentation masks at a logit level. The final segmentation mask is obtained by thresholding the fused mask.

that serve as prompts for the zero-shot model, enabling it to generate a refined segmentation mask. The outputs of the two models are then fused at the logit level, and the resulting scores are thresholded to obtain the final binary mask. This pipeline leverages both the task-specific strengths of the base segmentator and the generalization capability of the zero-shot model, producing consistently improved segmentations.

In the following, we first illustrate the segmentators that we use, then we describe the methods we consider to generate prompts for the zero-shot segmentators.

5.3.1 2D Semantic Segmentation Models

DeepLabv3+ [21] utilizes a fully convolutional network structure, enabling end-to-end training and inference on arbitrary-sized images. The network architecture consists of an encoder-decoder structure that leverages atrous convolutions and atrous spatial pyramid pooling (ASPP) to capture multi-scale contextual information. It introduces a skip connection from the encoder to the decoder module to incorporate low-level details from early layers of the network and improve the localization accuracy of the segmented objects.

The pyramid vision transformer (PVT) [202] is a vision transformer network that doesn't apply convolutions. Its core concept is to acquire high-resolution representations from finely-detailed input. The network's depth is paired with a progressively narrowing pyramid, enabling a reduction in computational burden. Additionally, to further curtail computational overhead, the system incorporates a spatial-reduction attention layer.

The segment anything model (SAM) [86] is a vision foundation model specifically designed for promptable image segmentation. It has been trained on the extensive SA-1B dataset, which includes 11 million images and more than 1 billion masks, making it on of the largest segmentation dataset. This vast training set enables SAM with excellent zero-shot capabilities. SAM consists of three main modules: an image encoder, a flexible prompt encoder, and a fast mask decoder. The image encoder utilizes a vision transformer (ViT) backbone to process high-resolution 1024×1024 images and generate 64×64 image embeddings. The prompt encoder handles both sparse prompts (e.g., points, boxes, text) and dense prompts (e.g., masks) by converting them into c-dimensional tokens. Finally, the mask decoder combines the image and prompt embeddings to produce segmentation masks in real-time. We evaluated two versions of SAM: ViT-Huge (ViT-H) and ViT-Large (ViT-L). These vary in the complexity of the input image ViT-based encoder, with the former model having 632M parameters and the latter having 307M parameters.

The segment everything everywhere all-at-once model (SEEM) [244] is a promptable, interactive model for image semantic segmentation. It predicts semantic masks based on the

(A) (B) (C) (D)

FIGURE 5.2: Example of checkpointing using a segmentation mask overlayed on an image. (**a**) Method A places a checkpoint at the average coordinates of each blob. (**b**) Method B puts a checkpoint at the center of mass of each blob. (**c**) Method C randomly selects a point within each blob. (**d**) Method D provides a grid of checkpoints.

interactions between the input image and multi-modal prompts. To do this, it encodes points, masks, text, boxes, or even a similar referred region of another image into the same joint visual-semantic space. SEEM employs a generic encoder-decoder architecture. Learnable queries interact with visual, text, and memory prompts through a self-attention mechanism. It was trained on COCO2017 [109] with panoptic segmentation annotations.

5.3.2 Checkpoint Engineering

We devised several strategies to generate checkpoints (prompts) aimed at improving prompt-based segmentators. We want to test whether specific prompt generation methods can enhance segmentation performance.

We designed four checkpoint generation methods, denoted as A-D:

A computes the average coordinates of a blob. This is simple but may place checkpoints outside the blob;

B computes the blob's center of mass, reducing the likelihood of placing checkpoints outside the blob;

C randomly selects a point within the blob, ensuring diversity and efficiency;

D generates multiple checkpoints via a uniform grid of step size b in both x and y. Checkpoints lying inside the blob are retained. A modified version, border mode (bm), applies morphological erosion using a 10×10 elliptical kernel before checkpoint extraction, reducing noisy boundaries If no checkpoints are found (e.g., due to small or irregular blobs), a fallback strategy shifts the grid horizontally and vertically until overlap occurs. At least one checkpoint is extracted per blob, except for trivial blobs (<10 pixels). The pseudo-code for Method D with fallback is reported in Algorithm 3.

Examples of the resulting checkpoints are shown in Figure 5.2.

Algorithm 3: Method D with mask erosion and fallback strategy.

Input:
mask ; // segmentation mask from which to sample the checkpoints
dp ; // grid sampling step
es ; // erosion size
Result: checkpoints
checkpoints ← empty list ; // List to store the selected checkpoints
dx = dy = 0 ; // offsets along x and y directions

/* count the number of non trivial blobs in mask */
blobs_num = count_blobs(mask) ;

while *checkpoints is empty AND dy < dp* **do**
  /* create a uniformly spaced grid of checkpoints with step dp, horizontal offset dx and vertical offset dy */
  grid ← create_uniform_grid(dp, dx, dy) ;
  while *checkpoints is empty AND es > 0* **do**
    /* erode mask with elliptical-shaped kernel of size of 10x10 px */
    eroded_mask ← erode_mask(mask, 10) ;
    checkpoints ← grid ∩ eroded_mask ;
    /* count the number of blobs having at least a checkpoint inside */
    extracted_blob_idxs = count_blobs_spanned(checkpoints, mask) ;
    if *extracted_blob_idxs ≠ blob_idxs* **then**
      checkpoints ← empty list ;
      es ← *es* − 1
    end
  end
  dx ← *dx* + 10 ;
  if *dx* ≥ *dp* **then**
    dx ← *dx* mod *dp* ;
    dy ← *dy* + 10 ;
  end
end

5.4 Experimental Setup

5.4.1 Datasets and Metrics

We evaluated our approach on seven heterogeneous datasets: CAMO, Portrait, Locust-mini, VinDr-RibCXR, SKIN, Butterfly, and a subset of COCO2017. Each dataset offers unique characteristics and challenges, allowing a comprehensive evaluation. For all datasets, we adopt the official train/test splits reported in the literature, except for COCO2017, where a custom split is defined.

The CAMO dataset [97] contains 1250 natural images (1000 train / 250 test) with camouflaged objects, providing complex textures and occlusions that mimic real-world conditions.

The Portrait dataset [83] includes 1736 portrait images (1447 train / 289 validation), designed to evaluate segmentation of human features and backgrounds.

The Locust-mini dataset [111] consists of 994 images (874 train / 120 test) of camouflaged locusts and grasshoppers, characterized by fine texture and color variability.

The VinDr-RibCXR dataset [134] provides 245 chest X-ray scans (196 train / 49 test) for rib segmentation, used here to test generalization to medical imaging tasks.

The SKIN dataset [118] aggregates multiple skin segmentation datasets.

The Butterfly dataset [198] includes natural images of butterflies with high intra-class variation; we follow the standard 4-fold cross-validation protocol with 624 training and 208 test images per fold, resizing all inputs to 513×513.

Finally, from COCO2017 [109], we extract a binary subset, COCO_animals, by selecting all images labeled under the "animal" supercategory. This yields 23,977 training and 1016 validation samples with background/animal annotations.

In all datasets we adopted two commonly used metrics: intersection over union (IoU) and Dice similarity coefficient (Dice).

IoU has already been discusses in previous chapters (see Chapter 4).

Dice [173] is defined as

$$\text{Dice}(P, T) = \frac{2|P \cap T|}{|P| + |T|} \tag{5.1}$$

and it measures the overlap between the predicted segmentation mask and the ground truth mask.

5.4.2 Baseline Extraction

The baseline performance in our experiments is established by evaluating the results of the DeepLabv3+ model, which was trained end-to-end on each of the datasets in this study. In addition, the PVTv2 segmentator ensemble is applied to the CAMO dataset.

We employed a DeepLabv3+ model with ResNet101 as the backbone architecture, training pre-trained weights on Pascal VOC2012 Aug dataset [39].

DeepLabv3+'s hyperparameters are an initial learning rate of 0.01, a total of 10 epochs for training, a momentum value of 0.9, L2 regularization with a coefficient of 0.005, a learning rate drop period of 5 epochs, a learning rate drop factor of 0.2, and stochastic gradient descent optimizer. We employed three data augmentations: horizontal flip, vertical flip, and 90° rotation.

We use an ensemble of six PVT networks, combined by average rule, constructed as follows:

- we apply two different data augmentation, defined in [133]: DA1, a base data augmentation consisting in horizontal and vertical flip, 90° rotation; DA2, which applies a rich set of diverse operations to derive new images from the original ones. These operations encompass shadowing, color mapping, vertical or horizontal flipping, and others.
- we apply three different learning strategies: learning rate of 1×10^{-4}; learning rate of 5×10^{-4} decaying to 5×10^{-5} after 10 epochs; learning rate of 5×10^{-5} decaying to 5×10^{-6} after 15 epochs and to 5×10^{-6} after 30 epochs.

The six networks are obtained by coupling the two DA methods with the three strategies to determine the learning rate. Each PVT network is trained for 50 epochs with a batch size of 8 and AdamW [116] optimizer.

model	src mask	CAMO		Portrait		Locust-Mini		VinDr-RibCXR	
		IoU	Dice	IoU	Dice	IoU	Dice	IoU	Dice
DLV3+ (baseline)	-	60.6	71.8	97.0	98.5	74.3	83.0	**63.5**	**77.6**
SAM ViT-L (D50bm)	DLV3+	60.9	70.7	95.8	97.8	59.7	70.2	28.0	43.6
SAM ViT-H (D50)	DLV3+	59.8	69.6	95.7	97.8	59.5	69.9	28.0	43.5
SEEM (D10bm)	DLV3+	54.2	62.1	95.9	97.9	57.3	65.6	32.1	48.4
Fusion SAM (ViT-H, D30)	DLV3+	**63.9**	**74.4**	97.1	98.5	**76.0**	**84.2**	61.0	75.6
Fusion SAM (ViT-H, D50)	DLV3+	63.0	73.4	**97.2**	**98.6**	75.7	84.1	60.7	75.4
Fusion SAM (ViT-H, D30bm)	DLV3+	63.5	74.2	97.1	98.5	**76.0**	**84.2**	61.0	75.6
Fusion SAM (ViT-H, D50bm)	DLV3+	63.0	73.5	**97.2**	98.5	75.7	84.1	60.7	75.4
PVTv2 (baseline)	-	71.8	81.1	-	-	-	-	-	-
SAM ViT-L (D50bm)	PVTv2	63.0	72.7	-	-	-	-	-	-
SAM ViT-H (D50)	PVTv2	69.8	78.5	-	-	-	-	-	-
SEEM (D10bm)	PVTv2	53.0	60.8	-	-	-	-	-	-
Fusion SAM (ViT-H, D30)	PVTv2	73.3	82.0	-	-	-	-	-	-
Fusion SAM (ViT-H, D50)	PVTv2	**73.5**	**82.0**	-	-	-	-	-	-
Fusion SAM (ViT-H, D30bm)	PVTv2	73.1	81.9	-	-	-	-	-	-
Fusion SAM (ViT-H, D50bm)	PVTv2	73.3	81.9	-	-	-	-	-	-

TABLE 5.1: Comparison of segmentation performance (IoU/Dice) across all datasets.

5.4.3 Fusion Step

To improve segmentation quality, we introduce a refinement step that fuses the logit masks from SAM with those of DeepLabv3+. For some datasets, we additionally fuse SAM with PVTv2 to benchmark against SOTA performance. The fusion is performed with a weighted rule, where pixel-wise logits are combined and thresholded to obtain the final binary mask. In our setup, the specialized segmentator (DeepLabv3+/PVTv2) is assigned a weight of 2 to balance contributions.

This strategy exploits the complementarity of different models, mitigates noise and biases in single-model predictions, and enhances boundary localization through consensus. By combining logits rather than masks, the method preserves fine details while reducing inconsistencies.

Implementation details include scaling all outputs to the [0,255] range and storing them in grayscale JPEG format for space efficiency. As JPEG compression thresholds negatives to zero, no further adjustments are required. Finally, to align conventions, SAM outputs (where 0 denotes background and 255 foreground) are inverted before fusion to match the coding of DeepLabv3+/PVTv2.

5.5 Experimental Evaluation

5.5.1 Results

In this section, we present and discuss the results of our experiments, summarized in Tables 5.1–5.2.

From the first group of experiments (in Table 5.1 we report the best performing methods, using the CAMO, Portrait, Locust-mini, and VinDr-RibCXR datasets), we observe that SEEM proves less promising than SAM under the considered prompting strategies and datasets, which led us to exclude SEEM from the second group of experiments.

Across datasets and fusion-less methods, neither SAM (ViT-L, ViT-H) nor SEEM surpass DeepLabv3+, regardless of the prompting method. Methods A–C never achieve top performance, thus we did not report them for better clarity.

Dataset	Metric	DLV3+	PVTv2	DLV3+ ⓐ PVTv2	DLV3+ ⓦ PVTv2	SAM ⓦ DLV3+	SAM ⓦ PVTv2
CAMO	**IoU** ↑	60.63	71.75	69.32	71.23	62.95	**73.46**
	Dice ↑	71.75	81.07	79.16	80.60	73.36	**82.02**
Portrait	**IoU** ↑	97.01	-	-	-	**97.18**	-
	Dice ↑	98.46	-	-	-	**98.55**	-
Locust-mini	**IoU** ↑	74.34	-	-	-	**75.74**	-
	Dice ↑	83.01	-	-	-	**84.07**	-
VinDr-RibCXR	**IoU** ↑	**63.48**	-	-	-	60.65	-
	Dice ↑	**77.57**	-	-	-	75.41	-
SKIN	**IoU** ↑	78.84	80.84	81.35	**81.50**	79.73	80.74
	Dice ↑	87.67	88.99	89.33	**89.44**	88.25	88.92
Butterfly	**IoU** ↑	95.47	97.00	96.82	96.96	96.02	**97.02**
	Dice ↑	97.66	98.47	98.37	98.45	97.94	**98.48**
COCO animals	**IoU** ↑	66.04	-	-	-	**69.15**	-
	Dice ↑	75.93	-	-	-	**77.93**	-

TABLE 5.2: IoU and Dice summary results computed by averaging various subsets within each dataset used in this work.

Within method D, no single parameter setting consistently yields the best results. For instance, CAMO benefits from a smaller sampling step b, whereas Portrait performs better with larger b values, producing fewer checkpoints. Mask erosion occasionally improves performance (e.g., VinDr-RibCXR), though differences are often minor and within noise levels.

The same phenomenon happen with segmentation masks produced by PVTv2. As a general choice, SAM ViT-H with $b = 50$ without mask erosion provides a good trade-off and is adopted in subsequent experiments.

We report in Table 5.2 a general comparison across methods and datasets, computing also the fusion between DeepLabV3+ and PVTv2 segmentation logits. We use the symbols ⓐ to indicate a fusion with the average rule (same weight for both masks), ⓦ to indicate a fusion with the weighted rule, and − to indicate no fusion at all, meaning that we used SAM's output using the segmentation mask of second term, the segmentator, for checkpointing. We used weight 1 for the first term, and weight 2 for the second. So for example, in SAMⓦDLV3+ the segmentation mask of DLV3+ is weighted 2, SAM's is weighted 1.

Comparing models, the most consistent gains come from the refinement strategy described in Section 5.4.3. Fusion with SAM, whether combined with DeepLabv3+ or PVTv2, consistently outperforms the baselines, in many cases by a substantial margin, except for VinDr-RibCXR dataset.

The strongest results are obtained on CAMO. Using DeepLabv3+ masks to guide SAM, SAM surpasses DeepLabv3+ itself. More importantly, fusing SAM with the PVTv2 ensemble outperforms PVTv2 alone, which has become the SOTA at the time this work was carried out.

Additional improvements are observed on SKIN and Butterfly. On SKIN, fusion (SAMⓦDLV3+ or SAMⓦPVTv2) consistently beats the corresponding baselines across many subsets. On Butterfly, SAM fusion slightly but consistently surpasses the strong SOTA ensemble (PVTv2

FIGURE 5.3: Examples from the SKIN (first and second rows) and Butterfly datasets.

+ CNN-based models). On COCO_animals, SAM's already competitive performance further benefits from fusion with DeepLabv3+. Figure 5.3 shows qualitative examples on SKIN and Butterfly datasets, Figure 5.4 on COCO2017 dataset, Figure 5.5 and Figure 5.6 on CAMO.

As a side note, we observed that modifying the fusion weight parameter (e.g., increasing the segmentator's weight from 2 to 3) produces slightly better IoU and Dice scores. A systematic study of this parameter is left as future work.

5.6 Conclusion

Our study shows that zero-shot segmentators such as SAM can enhance semantic segmentation when combined with specialized models like DeepLabv3+ and PVTv2. While SAM alone does not consistently outperform these models, checkpoint prompting and, especially, logit-level fusion yield clear improvements across multiple datasets, even surpassing state-of-the-art methods on CAMO and Butterfly.

We also observed limitations: performance varies across domains, prompts sometimes produce inconsistent or semantically weak outputs, and medical image segmentation remains particularly challenging. Ground-truth inaccuracies further complicate evaluation but also highlight the robustness of SAM in some cases.

Overall, fusing SAM with established models proves an effective and computationally inexpensive strategy, offering resilience to imperfect prompts and potential for state-of-the-art results. Future work should explore more diverse domains, alternative prompting strategies (e.g., bounding boxes, text), and tuning of fusion parameters to maximize performance.

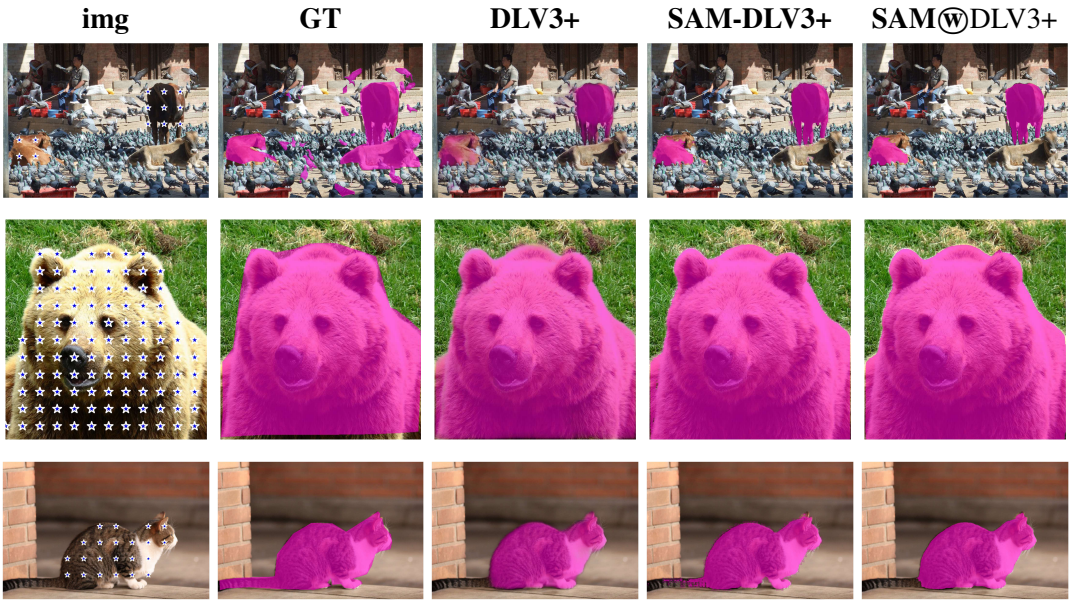
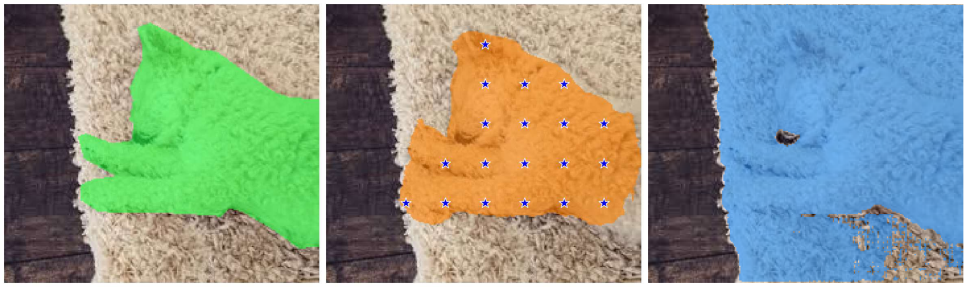


FIGURE 5.4: Examples from the COCO 2017 dataset.



(A) (B) (C)

FIGURE 5.5: A failure mode of SAM-DLV3+ on the CAMO dataset: despite strong prompting, SAM-DLV3+ fails to segment a common pet. (**a**) Ground truth. (**b**) Mask from DeepLabv3+ and corresponding checkpoints extracted by method D ($b = 30$, no mask erosion). (**c**) The output of SAM-DLV3+ when prompted with the aforementioned checkpoints

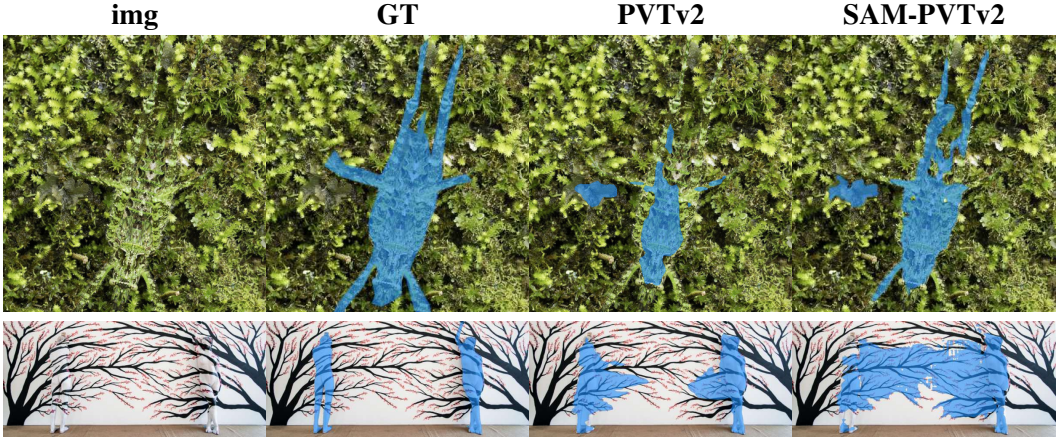


FIGURE 5.6: Examples from the CAMO dataset. The first two rows demonstrate significant improvements in segmentation, while the last two rows illustrate instances where the fusion process did not yield the desired results.

Chapter 6

EastPark: A Multi-Sensor, Large-Scale Dataset and PEFT Study for Efficient 3D LiDAR Semantic Segmentation

OBOTIC navigation in unstructured outdoor environments requires both semantic understanding of objects and detailed terrain characterization for safe traversal. Existing LiDAR datasets primarily target structured urban scenes and do not capture the complexity of crowded public spaces with heterogeneous ground surfaces. We introduce EastPark, a multi-sensor, large-scale dataset collected in a public park using a pitched sensor setup, designed as a benchmark for semantic segmentation in complex park environments. To establish baselines, we adapt several state-of-the-art 3D semantic segmentation models using parameter-efficient fine-tuning (PEFT), such as linear probing and low-rank adaptation (LoRA). The dataset and trained models provide a resource for studying semantic segmentation, traversability analysis, terrain characterization, and efficient adaptation techniques in robotic perception, and can be downloaded from https://zenodo.org/records/17121785.

6.1 Motivation

Light detection and ranging (LiDAR) has become a cornerstone sensing modality for robotic perception, enabling accurate 3D scene understanding across diverse tasks such as semantic segmentation, object detection, and traversability analysis [129, 243, 214, 45, 208]. Over the past years, large-scale LiDAR datasets such as SemanticKITTI [6], nuScenes [15], and Waymo[177] have accelerated progress in autonomous driving, providing well-curated benchmarks for perception in structured urban environments. However, these datasets are primarily designed for road traffic scenarios, with a strong emphasis on vehicles and traffic infrastructure, and they lack detailed coverage of terrain and people-rich environments that are equally critical for safe navigation.

To address this gap, we introduce a new multi-sensor, large-scale LiDAR, inertial measuring unit (IMU) and camera dataset captured in a public park using a pitched Ouster OS1-64 3D LiDAR with integrated IMU and a Kinetic Azure RGB-D camera. The pitched configuration provides enhanced ground coverage, which is crucial for characterizing terrain categories. The dataset contains more than 9,000 full 3D scans, each manually annotated by multiple expert annotators into 22 semantic categories, and 13000 RGB-D images. As shown in Fig. 6.1, the accumulated LiDAR scans depict the variety of semantic classes across the

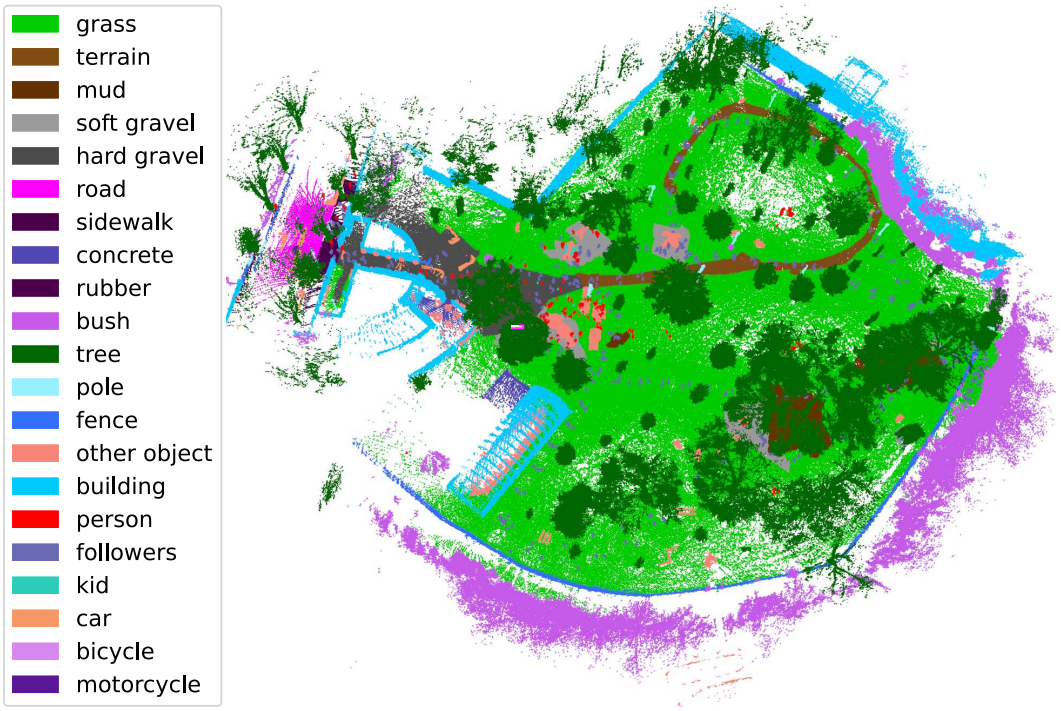


FIGURE 6.1: Global map of the dataset obtained by accumulating all LiDAR scans (with step 50). Colors indicate semantic annotations, with the legend showing each class and its corresponding color.

entire dataset. In Fig. 6.2, we provide qualitative examples from our dataset of synchronized camera images and annotated 3D LiDAR point clouds.

We establish baselines by fine-tuning state-of-the-art 3D segmentation models using parameter-efficient techniques, such as linear probing and low-rank adaptation (LoRA). The dataset, annotations, evaluation scripts, and trained models are publicly released to support research on terrain-aware perception, multi-modal fusion, and efficient adaptation methods for robotic systems operating in unstructured, people-rich outdoor environments.

6.2 Related Work

6.2.1 Public UGV Datasets

3D LiDAR semantic segmentation is essential for autonomous robot perception, supporting point-wise environmental understanding and safe navigation. Existing on-road datasets such as SemanticKITTI [6], SemanticPOSS [139], nuScenes [15], Pandaset [217], and Waymo Open Dataset [177] provide benchmarks in structured traffic environments. UGV-collected datasets in structured areas include NCLT [18], captured by a Segway UGV on a university campus, and M2DGR [226], using an AgileX Scout Mini for indoor and outdoor campus navigation.

For unstructured environments, UGV-based datasets are critical, as models trained on structured scenes perform poorly in irregular terrains with diverse vegetation and natural obstacles [130, 191]. RELLIS-3D [73] provides dense full-scene LiDAR, camera, and IMU data from off-road campuses. The Botanic Garden dataset [113] addresses park-like environments but only provides camera-based semantic labels. Specialized datasets like RUGD [206] and YCOR [123] target rugged ground classification and trail navigation, respectively.

Despite these contributions, a gap remains for large-scale, multi-sensor datasets in diverse public parks combining trails, wilderness, and heterogeneous vegetation. Our dataset

TABLE 6.1: Comparison of different datasets for navigation and perception. The top section shows datasets in structured environments, while the bottom section focuses on unstructured environments relevant for off-road UGV navigation.

		Environment				Stereo Vision / Camera				
Dataset	**Year**	**Scene**	**Type**	**Platform**	**LiDAR**	**Depth**	**RGB**	**IMU**	**3D Semantics**	**Scale**
Structured Environments (On-Road / Campus)										
SemanticKITTI [6]	2019	Urban	Struct	Vehicle	✓	-	✓	✓	Dense	Large
nuScenes [15]	2020	Urban	Struct	Vehicle	✓	-	✓	✓	Dense	Large
Waymo Open [177]	2020	Urban	Struct	Vehicle	✓	-	✓	✓	Dense	Large
Pandaset [217]	2021	Urban	Struct	Vehicle	✓	-	✓	✓	Dense	Large
Unstructured Environments (Off-Road / Natural)										
YCOR [123]	2018	Forest/Trail	Unstruct	UGV	-	-	✓	-	-	Small
RUGD [206]	2019	Off-road	Unstruct	UGV	✓	-	✓	✓	-	Small
Rellis-3D [73]	2020	Off-road	Unstruct	UGV	✓	✓	✓	✓	Dense	Large
Goose [130]	2024	Off-road	Unstruct	Vehicle	✓	-	✓	✓	Dense	Large
BotanicGarden [113]	2024	Natural Park	Unstruct	UGV	✓	-	✓	✓	-	Small
WildScenes [191]	2025	Forest	Unstruct	Human-operated	✓	-	✓	✓	Dense	Large
EastPark(Ours)	2025	**Natural Park**	**Unstruct**	**UGV**	✓	✓	✓	✓	**Dense**	**Large**

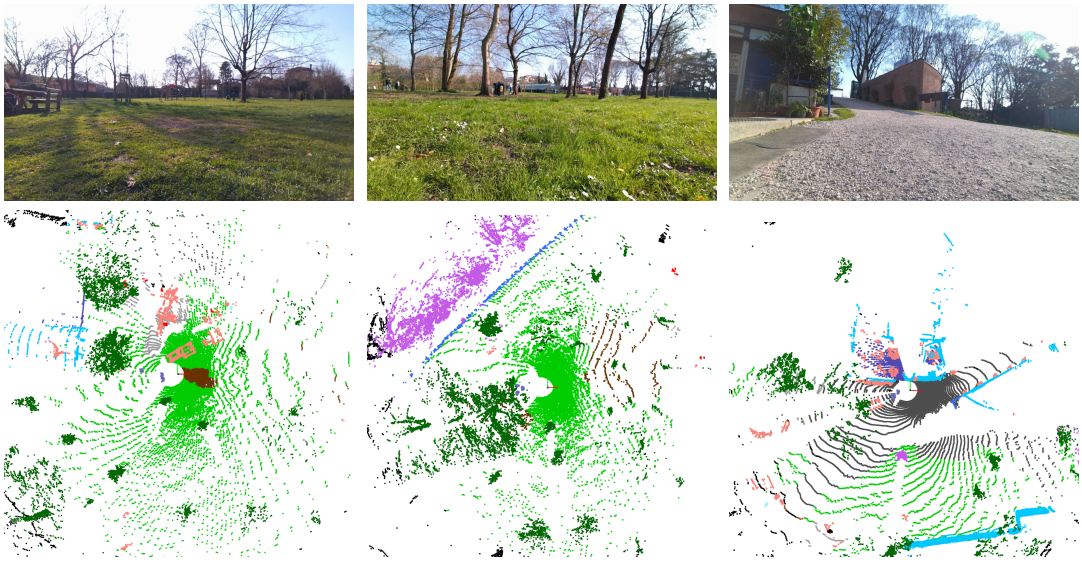


FIGURE 6.2: Examples from our dataset. Top: RGB images captured with Kinect Azure. Bottom: corresponding annotated 3D LiDAR point clouds. Each column shows a synchronized pair collected from diverse viewpoints around the park.

addresses this gap by providing high-fidelity 3D LiDAR scans with fine-grained manual annotations, offering a valuable benchmark for advancing mobile robotics perception.

6.2.2 3D LiDAR Semantic Segmentation Methods

Recent methods in 3D LiDAR semantic segmentation can be categorized into three main approaches: point-based, voxel-based, and projection-based. For a comprehensive review, see Chapter 4. In this work we focus on the following, recent semantic segmentation models.

Point Transformer v3 (PTv3) [214] is a point-based method that scales the transformer architecture to larger receptive fields. It uses an optimized self-attention mechanism to efficiently model long-range dependencies, achieving state-of-the-art performance on large-scale 3D semantic segmentation tasks.

LSK3DNet [43] improves voxel-based segmentation by using large sparse 3D convolutional kernels. It employs dynamic sparsity and channel pruning to optimize kernel structure during training, achieving efficient large-kernel operations with expanded receptive fields.

3PNet [131] is a hybrid projection-based method that combines the speed of 2D range image processing with local point-based features. This design maintains geometric detail while benefiting from computational efficiency, making it effective in low-data scenarios.

6.2.3 Efficient Fine-Tuning approaches

Adapting large pre-trained 3D models presents a critical trade-off. Linear probing for feature extraction is efficient but limits performance by ignoring rich intermediate features [182]. Conversely, full fine-tuning achieves higher accuracy but it requires learning unique parameters for each task and dataset, resulting in significant memory and storage demand [232]. This efficiency-performance gap has motivated the recent and active exploration of parameter-efficient fine-tuning (PEFT) within the 3D domain.

This exploration, evidenced by numerous recent studies [231, 240, 182, 42, 104, 183, 175, 41, 199], is dominated by two main strategies. The first involves modifying a small subset of the model's weights, including adapter-based approaches and the widely-adopted low-rank adaptation (LoRA) [68]. This has led to 3D-specific variants like PointLoRA [199], which is a direct evolution of the original LoRA, specifically adapted for the token-based structure of point cloud transformers for indoor environments.

However, a critical limitation unites these pioneering works: their validation is predominantly on clean, object-centric benchmarks such as ModelNet40 for classification or ShapeNet for part segmentation. The performance of these diverse PEFT strategies, ranging from LoRA and its variants to dynamic prompting, on the noisy, large-scale data from 3D LiDAR semantic segmentation remains largely unverified. The present study addresses this gap by presenting the first study to benchmark these methods in this challenging, real-world domain.

6.3 Dataset Description

6.3.1 Data Collection

We collected the data in March 2025 in a 150-meter-wide square green space. The heterogeneous terrain of the park, which includes trails, mud, and gravel, offers an ideal benchmark for key challenges in mobile robotics. The park presents a complex perceptual challenge, containing a dense mixture of natural vegetation (trees, bushes) and varied man-made structures such as a restaurant, playground equipment, benches, a monument, and a large ancient boat, making it highly suitable for developing and evaluating algorithms for SLAM, object detection, and semantic scene understanding in realistic, human-centric spaces. Fig. 6.3 provides a bird's-eye-view visualization of the acquired trajectories, projected onto a satellite image of the park.

The data acquisition platform is a four-wheeled differential-drive unmanned ground vehicle (UGV). We rigidly mounted a multi-sensor suite consisting of an Ouster OS1-64 3D LiDAR with integrated inertial measurement unit (IMU) and a Kinect Azure depth camera. During data collection, we recorded synchronized sensor streams from the LiDAR at 10 Hz, the IMU at 100 Hz, and the camera at 15 Hz.

The sensor configuration in our dataset differs from typical public datasets: both the LiDAR and camera are pitched downward rather than mounted horizontally (parallel to the robot base or the ground). This orientation allows the UGV to better capture the terrain directly ahead and improves perception robustness during navigation. Specifically, the LiDAR and camera are pitched by approximately 10.2° and 13° relative to the ground, and mounted at heights of 0.92 m and 0.85 m, respectively. Notably, these sensor heights and orientations

differ from those commonly found in other outdoor semantic segmentation datasets, providing a unique perspective on ground features.

6.3.2 Dataset Specifications and Structure

The EastPark dataset consists of 9056 annotated 3D point clouds, where each cloud corresponds to a full 360° scan from the LiDAR sensor, and 13584 RGB-D images. Point clouds are provided in a raw per-point format, including Cartesian coordinates (*x, y, z*), *intensity* and *timestamp*. Each point is further augmented with *reflectivity*, calibrated value indicating the material properties and texture of the scanned surface, *ring*, the index of the vertical laser channel that generated the point and *ambient*, a measurement of ambient near-infrared light, which can be used to identify and filter noise from external light sources like sunlight.

We collected four sequences, with 3120, 1184, 2607, and 2145 point clouds obtained for each, respectively. Although the initial three sequences are quite similar in terms of label distribution, the fourth sequence exhibits a higher degree of heterogeneity. This is due to the fact that the UGV was partially driven over a sidewalk along the road that limits the park, resulting in the encounter of classes that are not observed in the other sequences. Consequently, the fourth sequence was randomly partitioned into three smaller sequences, with the objective of ensuring that the class distribution was analogous in all three newly created partitions. Finally, the dataset consists of six sequences totaling 3120, 1184, 2607, 1073, 429, and 643 point clouds. Sequences 0 and 3 (4193 scans, 46%) are used for training, sequences 1 and 4 (1613 scans, 18%) for validation, and sequences 2 and 5 (3250 scans, 36%) for testing. Since LiDAR and camera have different frame rates, we provide RGB-D images with timestamps synchronized to the LiDAR and IMU data, allowing straightforward temporal alignment across all sensors for multi-modal perception tasks.

We provide the dataset with labels in two formats. The full version, *EastPark-full*, refers to the six-sequence dataset containing 21 classes (where follower is mapped to person), while "EastPark-mini" denotes the first three sequences with all labels mapped to a subset of 14 classes (see Tab 6.2).

6.3.3 Odometry Estimation

The raw LiDAR data provided has not been corrected for the motion distortion (skew) induced by the UGV's movement during scan acquisition. To generate a precise vehicle trajectory, we benchmarked several state-of-the-art LiDAR odometry and SLAM algorithms: GenZ-ICP [98], KISS-ICP [194], Kinematic-ICP [56], Lego-LOAM [163] and LIO-SAM [164].

We tested the aforementioned methods offline to obtain a robust odometry estimation. Consequently, we conducted a qualitative evaluation to determine the most effective odometry estimation method. GenZ-ICP [98] demonstrates the most accurate performance, producing point clouds with the fewest outlier poses compared to other methods. We validated this by constructing a global map through the accumulation of all point clouds aligned with the predicted poses.

6.3.4 Dataset Annotation

To ensure the integrity of the dataset annotations, we relied on the expertise of annotators actively involved in 3D point cloud processing and with prior experience with existing datasets. The dataset was meticulously reviewed to ensure it was as error-free as possible. To annotate the point clouds, we used the *point_labeler* tool [6], which enables labeling multiple point clouds at once for overlapping areas.

We annotated static classes on dense point clouds generated by accumulating motion-corrected scans, and we manually labeled dynamic classes on each individual scan.

FIGURE 6.3: Trajectories completed around the park, overlaid on a Google Satellite Image (©2025 Google). Sequence 00 ■, Sequence 01 ■, Sequence 02 ■, Sequence 03 ■ (best viewed in color).

Finally, we segmented 22 semantic labels: grass, terrain, mud, soft gravel, hard gravel, road, sidewalk, concrete, rubber, bush, tree, pole, fence, other object, building, person, followers, kid, car, bicycle, motorcycle, and unknown class. To exclude potentially unreliable measurements, we labeled all points beyond 30 m from the LiDAR sensor as unknown.

The follower class is particular: we annotated as follower the UGV drivers that closely follow the UGV. We did so in order for a possible future project to remove the points associated with this class, since in common operation, there may not be any followers. However, in our experiments (see Section 6.4), we remapped this class to the person class, actually obtaining 21 classes. The rubber class is a particular drivable surface that is found in the sidewalk near the park, especially near the trees.

The details of the label distribution are presented in Fig. 6.4.

6.4 Experimental Evaluation

The benchmark evaluation considers three state-of-the-art architectures: Point Transformer v3 (PTv3) [214], 3PNet [131], and LSK3DNet [43], representing transformer-based, range image-based, and voxel-based sparse kernel approaches, respectively. Official implementations and pretrained weights were used: PTv3 on nuScenes, 3PNet on SemanticKITTI and PandaSet, and LSK3DNet on SemanticKITTI and nuScenes.

We consider three evaluation configurations. Vanilla (zero-shot): pretrained models directly applied to EastPark, with label remapping. Linear probing: only the segmentation head is retrained on the new dataset, freezing all other parameters. LoRA fine-tuning: low-rank adaptation of selected model layers plus retraining the segmentation head, using PEFT with standard hyperparameters (rank=8, $\alpha = 16$, dropout=0.05) for all models. Specific layers adapted include transformer query/key/value/projection for PTv3, embedding and 1D conv layers for 3PNet, and `PPmodel`/`layer_in`/`layer_out` for LSK3DNet.

To improve generalization, we applied a common set of basic data augmentations to all models: random rotations around the z-axis, random flips along the x- and y-axes, and random scaling with a factor of ± 0.1. We adopted the same loss functions as defined in

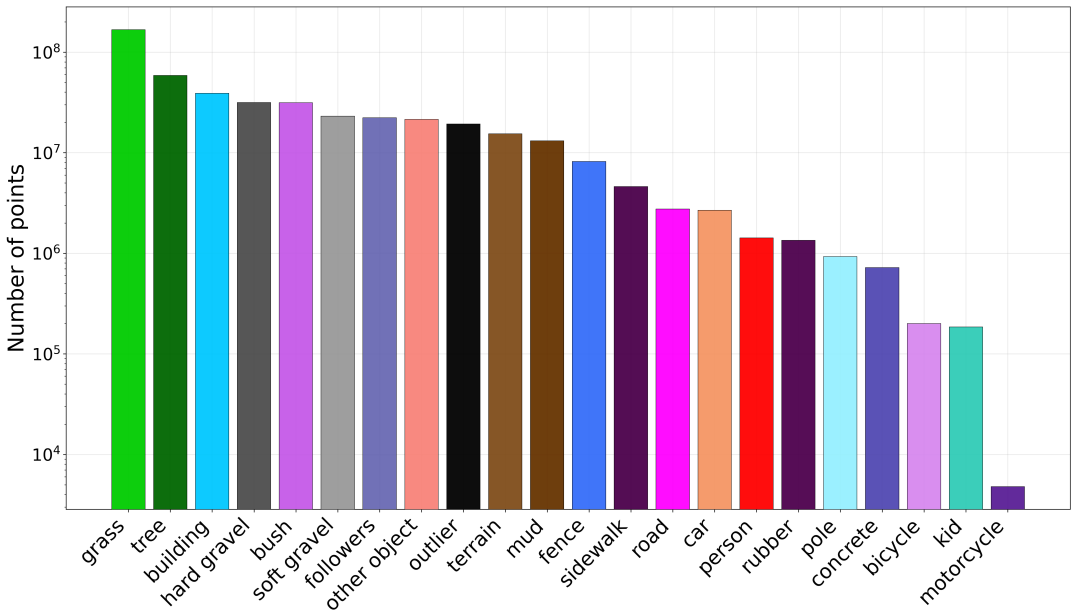


FIGURE 6.4: Distribution of labels by class in the EastPark dataset.

the original implementations of each method and we optimized using AdamW [116] with a learning rate of 1×10^{-3} and a weight decay of 1×10^{-4}, ensuring a fair comparison.

6.4.1 Evaluation protocol

Experiments are conducted under two configurations: *EastPark-mini*, a reduced version of the dataset with 14 semantic labels for efficient prototyping, and *EastPark-full*, the complete dataset with 21 labels.

We preprocessed all point clouds by rotating them to a normalized reference frame centered on the sensor such that the ground plane is horizontal.

6.4.2 Quantitative results

Tab. 6.2 reports results on the *EastPark-mini* setup, while Tab. 6.3 reports results on *EastPark-full*.

In general, zero-shot performance is low across all methods, confirming a substantial domain gap between road datasets and the park environment. Linear probing provides consistent improvements, but LoRA fine-tuning achieves significantly higher gains while updating only a fraction of the parameters.

In the mini configuration, 3PNet pre-trained on SemanticKITTI achieves the best performance in the zero-shot evaluation, whereas the model pre-trained on PandaSet performs best under linear probing. This suggests that the pre-trained backbones capture meaningful and transferable features, though their effectiveness depends on the adaptation strategy. LSK3DNet fine-tuned with LoRA, especially fine-tuning the nuScenes pre-trained model, consistently achieves the highest performance across configurations, highlighting the effectiveness of sparse kernels and PEFT.

In the full configuration, PTv3 pretrained on nuScenes shows the best zero-shot performance, and 3PNet on PandaSet performs best under linear probing, but LSK3DNet with LoRA still outperforms all methods.

These results highlight three key insights:

Method	Pre-Train	Model	mIoU	grass	terrain	mud	soft gravel	hard gravel	concrete	bush	tree	pole	fence	other object	building	person	% params
PTv3	nuScenes	Vanilla	4.8	6.5	7.2	†	†	†	†	†	†	†	†	0.9	14.1	33.7	0
3PNet	SemKITTI	Vanilla	12.7	0.1	6.7	†	†	†	†	†	19.4	17.5	44.8	0.1	3.0	73.5	0
3PNet	Pandaset	Vanilla	0.3	0.4	1.0	†	†	†	†	†	†	†	†	1.6	0.0	1.2	0
LSK3DNet	SemKITTI	Vanilla	6.1	4.8	3.9	†	†	†	†	†	9.4	0.0	1.0	6.0	0.1	54.5	0
LSK3DNet	nuScenes	Vanilla	2.6	25.0	7.8	†	†	†	†	†	†	†	†	0.5	0.2	0.5	0
PTv3	nuScenes	Linear	29.7	53.2	1.4	7.9	16.0	1.7	0.1	37.7	60.3	10.0	36.1	26.4	69.0	66.3	ϵ
3PNet	SemKITTI	Linear	41.7	77.1	1.4	8.0	27.1	0.5	0.3	65.1	67.0	53.7	59.3	32.9	66.0	83.3	0.1
3PNet	Pandaset	Linear	42.2	76.0	0.4	24.7	19.2	3.9	1.2	61.2	73.6	64.8	52.8	37.9	57.2	75.6	0.1
LSK3DNet	SemKITTI	Linear	29.8	74.8	2.2	3.8	10.3	0.0	0.0	43.7	43.8	29.9	38.4	24.9	38.7	77.6	ϵ
LSK3DNet	nuScenes	Linear	31.0	74.7	1.5	5.7	3.6	3.6	0.1	39.1	50.1	9.7	30.9	29.7	61.1	93.7	ϵ
PTv3	nuScenes	LoRA	56.0	77.8	31.3	47.1	39.2	1.8	0.3	68.1	80.6	69.4	**70.3**	62.2	86.6	93.7	0.4
3PNet	SemKITTI	LoRA	49.2	79.4	12.4	26.5	48.8	2.4	0.3	40.4	65.3	64.9	69.1	54.6	81.3	94.2	6.0
3PNet	Pandaset	LoRA	55.4	80.7	8.3	49.9	35.1	6.2	3.0	75.6	**85.1**	78.4	68.7	55.4	76.7	96.6	6.0
LSK3DNet	SemKITTI	LoRA	57.2	81.7	31.9	54.8	52.1	2.7	0.4	71.1	71.9	69.2	67.2	62.7	85.3	93.0	0.1
LSK3DNet	nuScenes	LoRA	**64.9**	**84.6**	**35.9**	**70.9**	**57.2**	**11.8**	**7.8**	**78.7**	83.2	**83.5**	68.0	**74.3**	**91.9**	**96.7**	0.1

TABLE 6.2: Quantitative comparison of semantic segmentation performance (percentage of mIoU, per-class IoU and number of trained parameters over total number of trainable parameters) across different methods, datasets, and training strategies in the *EastPark-mini* dataset version. † denotes labels that are absent in the original label set of a vanilla method. ϵ indicates a strictly positive number < 0.1.

- pretraining source matters, since models pretrained on nuScenes transfer better to our dataset than those pretrained on SemanticKITTI or PandaSet.
- PEFT is highly effective, with LoRA consistently improving upon linear probing without requiring full fine-tuning.
- sparse kernel approaches, i.e., LSK3DNet, are particularly well-suited to environments with fine-grained terrain and dense human activity.

6.5 Conclusion

In this chapter, we introduced EastPark, a new multi-sensor, large-scale dataset for semantic segmentation and terrain analysis in crowded park environments, with unique sensor placement and dense manual annotations of 3D LiDAR point clouds over 22 classes. To establish baselines, we evaluated three state-of-the-art models (PTv3, 3PNet, and LSK3DNet) using official pretrained weights and adapting them with parameter-efficient strategies.

Experiments on both the *EastPark-mini* and *EastPark-full* configurations show that (i) pretraining source strongly affects transferability, with nuScenes-pretrained models generalizing best; (ii) parameter-efficient fine-tuning, particularly LoRA, provides large gains over linear probing at minimal cost; and (iii) sparse kernel approaches such as LSK3DNet are especially effective in capturing fine-grained structures like vegetation, different types of terrain and trees, which dominate our dataset.

In future work, we plan to extend the benchmark to instance and panoptic segmentation, and to exploit the RGB-D camera and IMU modalities for multi-sensor fusion. Also, we plan to provide annotations for the camera images.

method	pre-train	model	mIoU	grass	terrain	mud	soft gravel	hard gravel	road	sidewalk	concrete	rubber	bush	tree	pole	fence	other object	building	person	kid	car	bicycle	motorcycle	% params
PTv3	nuScenes	Vanilla	5.2	5.2	6.1	†	†	†	6.2	4.3	†	†	†	†	†	†	0.7	26.3	36.9	†	17.2	†	0.0	0
3PNet	SemKITTI	Vanilla	2.7	0.1	5.9	†	†	†	1.6	0.2	†	†	†	2.8	9.2	9.6	0.0	2.2	21.0	†	0.7	0.0	0.0	0
3PNet	Pandaset	Vanilla	1.5	0.2	1.2	†	†	†	0.9	0.1	†	†	†	†	†	†	0.0	21.9	0.0	†	5.3	0.0	0.0	0
LSK3DNet	SemKITTI	Vanilla	3.6	4.0	2.7	†	†	†	7.6	0.2	†	†	†	10.0	0.0	0.9	4.6	0.3	41.4	†	0.0	0.0	0.0	0
LSK3DNet	nuScenes	Vanilla	1.4	15.2	6.3	†	†	†	0.4	2.0	†	†	†	†	†	†	0.3	0.9	1.9	†	1.2	†	0.0	0
PTv3	nuScenes	Linear	27.1	52.9	1.0	5.4	15.9	22.4	15.8	26.1	3.2	3.8	38.3	60.1	11.3	23.8	26.2	64.4	75.1	0.0	77.9	17.7	0.0	ϵ
3PNet	SemKITTI	Linear	36.9	73.7	0.1	1.1	16.5	33.2	36.9	38.7	21.3	17.8	56.1	61.5	43.0	51.6	27.5	66.9	77.6	2.3	78.6	28.1	5.9	0.1
3PNet	Pandaset	Linear	39.1	72.3	0.3	13.0	11.5	37.1	68.4	43.0	25.8	20.6	54.4	68.4	51.0	40.7	34.9	64.4	68.8	3.9	73.5	9.0	21.7	0.1
LSK3DNet	SemKITTI	Linear	26.5	69.6	0.0	0.0	8.9	16.5	5.2	22.5	5.2	0.5	43.7	40.1	31.6	24.4	18.5	53.7	77.9	0.0	64.9	47.1	0.0	ϵ
LSK3DNet	nuScenes	Linear	28.0	71.1	0.0	3.8	4.6	27.3	20.5	21.8	7.2	2.4	35.0	47.7	18.2	22.4	25.7	62.0	89.0	0.0	78.3	20.9	3.1	ϵ
PTv3	nuScenes	LoRA	61.2	80.6	16.6	49.1	54.3	70.1	74.9	69.1	47.3	45.3	67.3	82.2	70.0	**69.9**	75.7	89.4	96.6	0.4	90.6	75.0	0.0	0.4
3PNet	SemKITTI	LoRA	51.3	78.1	5.6	34.6	50.0	53.7	45.4	39.1	42.9	43.6	57.9	69.6	64.8	64.3	62.2	80.4	91.9	2.0	88.4	39.4	12.3	6.0
3PNet	Pandaset	LoRA	36.9	73.6	0.4	10.2	30.3	31.3	20.7	41.1	19.3	0.3	63.3	80.6	65.6	48.4	48.1	68.8	88.2	0.0	46.9	0.0	0.0	6.0
LSK3DNet	SemKITTI	LoRA	63.8	81.0	**32.6**	55.4	55.3	72.1	90.9	79.8	36.2	63.2	68.5	72.9	73.8	67.2	69.4	88.1	95.5	2.3	92.1	78.9	0.9	0.1
LSK3DNet	nuScenes	LoRA	**75.7**	**83.6**	26.0	**70.0**	**62.5**	**84.3**	**95.3**	**84.5**	**83.5**	**72.9**	**83.4**	**86.6**	**79.8**	68.7	**79.5**	**90.9**	**97.7**	**2.5**	**96.4**	**96.3**	**70.2**	0.1

TABLE 6.3: Quantitative comparison of semantic segmentation performance (percentage of mIoU, per-class IoU and number of trained parameters over total number of trainable parameters) across different methods, datasets, and training strategies in the *EastPark-full* dataset version. † denotes labels that are absent in the original label set of a vanilla method. ϵ indicates a strictly positive number < 0.1.

Chapter 7

Horticultural Temporal Fruit Monitoring via 3D Instance Segmentation and Re-Identification using Colored Point Clouds

ACCURATE fruit monitoring over time is a critical component of automated agricultural production, yet it remains challenging due to fruit variability, occlusions, and the dynamic nature of orchards. In this chapter, we present a method for fruit instance segmentation and re-identification on 3D terrestrial point clouds collected across multiple sessions. Our approach segments individual fruits directly from dense colored point clouds, extracts discriminative descriptors using a 3D sparse convolutional network, and employs an attention-based probabilistic matching scheme to track fruits over time. Experiments on real strawberry and apple datasets demonstrate that the method achieves superior performance in both segmentation and temporal re-identification, supporting robust and scalable fruit monitoring in complex orchard environments.

7.1 Motivation

Meeting the rising demand for food requires more efficient and sustainable agricultural practices. Autonomous systems play a key role by automating crop monitoring and phenotyping, enabling large-scale and high-throughput analysis of plant traits beyond the limits of manual observation [38, 195, 44, 205]. A critical step toward this goal is the ability to track individual fruits over time, which supports growth analysis and yield estimation.

The task of temporal matching, or fruit re-identification, requires solving two interdependent problems: (i) accurate fruit instance segmentation and (ii) reliable temporal re-identification across different scans. Unlike 2D imaging, 3D point clouds provide precise spatial and depth information but are irregular and sparse, making them challenging to process. Fruits further complicate re-identification, as they lack unique visual traits, can be tightly clustered, and may change pose, appearance, or even presence over time.

To address these challenges, we propose a method that directly operates on dense colored point clouds. Fruits are first segmented using a learning-based approach and represented by descriptors extracted with a 3D sparse convolutional neural network. An attention-based matching module then associates fruits across acquisition sessions, explicitly handling the no-match case for new instances.

We claim that (i) our approach effectively segments fruit instances from real 3D scans, and (ii) it achieves superior re-identification performance compared to baselines on real-world strawberry datasets. By leveraging 3D descriptors and attentive matching, the method

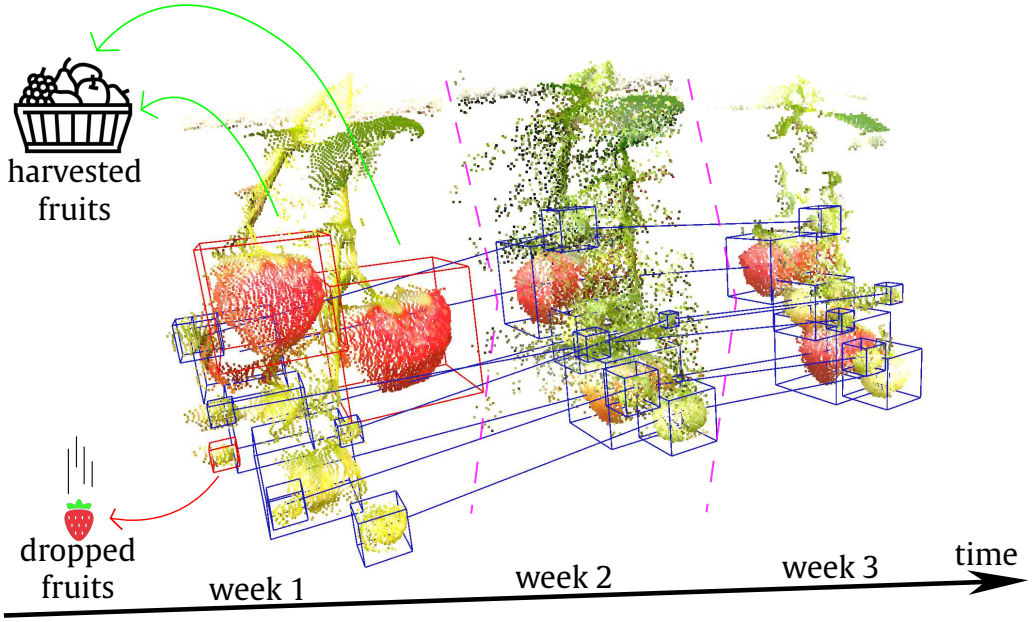


FIGURE 7.1: Fruit re-identification on three point clouds acquired at three different points in time. Fruit instances are first segmented using an instance segmentation method, then they are temporally matched with fruit instances of a previous data collection (e.g., matching fruits recorded in week t with fruits from week $t+1$). Blue bounding boxes (■) indicate matched fruit instances, while red bounding boxes (■) indicate unmatched fruits (e.g., harvested, dropped or newly appeared).

advances automated phenotyping and contributes to scalable monitoring in dynamic agricultural environments.

The implementation of our fruit matching method is publicly available at https://github.com/PRBonn/IRIS3D.

7.2 Related Work

Our work intersects several research areas: instance segmentation, plant phenotyping, and temporal object matching. We first review 2D and 3D instance segmentation methods, both general-purpose and agriculture-oriented. We then summarize plant phenotyping approaches in images and point clouds. Finally, we discuss temporal matching strategies for object tracking over time, highlighting their relevance for fruit monitoring.

Instance segmentation on 2D images : General instance segmentation, the task of segmenting individual objects within a scene, has been extensively studied in the context of 2D images. Among the most influential methods is Mask R-CNN [62], which extends object detection by predicting pixel-level masks. It has been successfully applied to fruit segmentation on images [146, 47, 53, 50]. [72] proposed an anchor-free variant designed to improve stability and facilitate transfer to unseen fruit types without hyperparameter tuning.

Instance segmentation on RGB-D images: While 2D image-based instance segmentation methods have shown promising results, they often struggle with occlusions and overlapping objects, which are common in agricultural environments. To address these challenges, some approaches have incorporated depth information from RGB-D images. Ge et al. [48] adopts Mask R-CNN [62] to segment and localize fruits in RGB-D images. Kang et al. [76]

propose DaSNet-v2, a multi-task network that jointly performs detection and instance segmentation on fruits, and semantic segmentation on branches, in RGB-D images collected in apple orchards. It applies feature pyramid networks and atrous spatial pyramid pooling to effectively capture multi-scale features and context information. Tang et al. [181] propose a high-precision apple instance segmentation method based on an improved SOLOv2 [203] and EfficientNet [179] backbone using RGB-D images. In scenarios involving overlapping or occluded apples, authors apply a lightweight spatial attention module to improve segmentation accuracy. Magistri et al. [120] exploit shape completion and differentiable rendering techniques to estimate the 3D shape of a target fruit together with its pose even under strong occlusions from a single RGB-D image. RGB-D images provide additional spatial information that can help disambiguate overlapping objects and improve segmentation accuracy in complex scenes, but they still lack the full 3D structural detail that point clouds can offer. The point clouds generated from RGB-D images are typically sparse and noisy, which can limit the effectiveness of 3D instance segmentation methods.

Instance segmentation on 3D point clouds : Instance segmentation on 3D data is considerably more challenging, as point clouds captured by RGB-D cameras or LiDARs are sparse and irregular. Most methods rely on 3D convolutional neural networks (CNNs) to learn features, often requiring large annotated datasets, which are scarce in agriculture. Data augmentation [241] alleviates this to some extent, but the lack of scale remains a bottleneck. A common strategy is to voxelize the point cloud and apply sparse convolutions [54], enabling efficient memory use. Recent advances include Mask3D [161], which leverages transformer decoders and instance queries; MaskPLS [122], optimized for autonomous driving with multi-scale sparse backbones; and the Superpoint Transformer [154], which combines handcrafted superpoints with graph-attention layers. Other innovations include Spherical Mask [167], which predicts 3D polygons via point-wise offsets for fine-grained clustering. Despite these advances, no prior work has directly addressed 3D fruit instance segmentation on point clouds. Closely related approaches combine modalities, e.g., fusing point clouds and images for fruit localization [77], or reconstructing occluded fruit geometry from RGB-D data using differentiable rendering [120].

Image and 3D plant phenotyping : Image-based phenotyping has become an essential tool in agriculture, using computer vision and deep learning to assess plant traits [156, 110, 27, 17, 80, 135, 60, 172, 11, 52]. Fewer studies operate on 3D data, though recent works demonstrate its potential: [61] measure wilting in cotton using point clouds, while [12] estimate internode lengths in cucumbers. The main challenge lies in sensor resolution: LiDARs and RGB-D cameras often lack sufficient density for agricultural tasks. High-precision terrestrial laser scanners, however, can generate dense, colored point clouds, as in [155], where a ground robot automates laser scanning for field phenotyping.

Temporal matching : Temporal fruit matching, i.e. associating fruits across multiple acquisitions, remains underexplored. [19] track plant growth by matching skeletal structures of leaves over time. [152] introduce a histogram descriptor based on spatial fruit distributions. More recently, [115] addressed 4D data association in greenhouse peppers by combining 3D RGB-D SLAM with visual place recognition; their follow-up work [114] improved this with deep descriptors and geometric constraints.

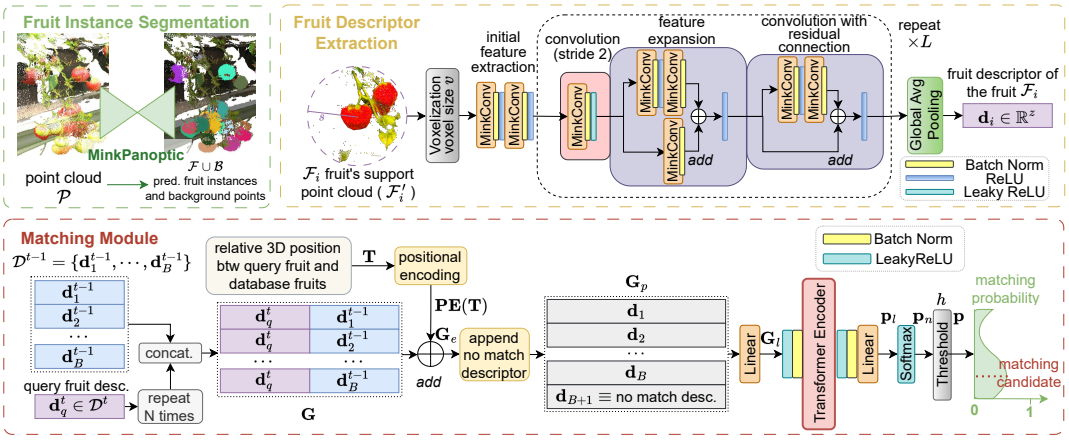


FIGURE 7.2: Pipeline of our approach. Fruit instance segmentation provides K fruit instance masks $\mathcal{F} = \{\mathcal{F}_i\}_{i=1}^{K}$ using a colored point cloud $\mathcal{P}$. The fruit descriptor extraction module processes a fruit's support point cloud, $\mathcal{F}_i'$, and computes the fruit descriptor, $\mathbf{d}_i$. It initially voxelizes the input point cloud, then, a MinkowskiNet [26] encoder processes the voxelized point cloud by leveraging sparse 3D convolutions. A final global average pooling aggregates the features of all voxels to compute the descriptor. The matching module matches a query descriptor $\mathbf{d}_i$ with a set of descriptors $\mathcal{D}^{t-1}$ of fruits that belong to a different point in time.

In contrast to prior work, we rely on MinkPanoptic (a core component module of MaskPLS [122]) for our instance segmentation method. Based on such a segmentation, we can address different downstream tasks, like plant and fruit phenotyping. We then extract compact descriptors using 3D sparse convolutions, which automatically learn discriminative features without handcrafted engineering and generalize to unseen data. Finally, we introduce an attention-based matching network that predicts probabilistic correspondences across time, explicitly modeling the no-match case.

7.3 Our Approach

We aim to track fruits using real 3D data acquired by a high-resolution LiDAR scanner. We present a novel method for accurately performing fruit instance segmentation and re-identification on point clouds captured at different points in time.

7.3.1 Fruit Instance Segmentation Module

Let $\mathcal{P}$ denote a colored point cloud where each point has 3D coordinates $\mathbf{p}_i \in \mathbb{R}^3$ and RGB value $\mathbf{k}_i$. The goal is to partition $\mathcal{P}$ into a set of K fruit instances $\mathcal{F} = \{\mathcal{F}_i\}_{i=1}^{K}$ and background points $\mathcal{B}$. For each fruit $\mathcal{F}_i$, we define its center $\mathbf{c}_i$ as the mean of its points:

$$\mathbf{c}_i = \frac{1}{|\mathcal{F}_i|} \sum_{\mathbf{p}_j \in \mathcal{F}_i} \mathbf{p}_j, \tag{7.1}$$

and the radius r_i of the smallest sphere, centered at $\mathbf{c}_i$, containing all the fruit's points.

We adopt MinkPanoptic [122], a learning-based instance segmentation method, selected because of its superior performance in our limited-data setting (see Section 7.4).

Let $\mathcal{P}_f \subseteq \mathcal{P}$ be the subset of $\mathcal{P}$ that was labeled as fruit. Then, for each point in $\mathcal{P}_f$, the method predicts a 3D vector representing the point's offset from the center of the fruit instance it belongs to. We add the predicted offsets to $\mathcal{P}_f$ to get the point cloud $\mathcal{P}_o$. We

determine fruit instances $\mathcal{F}$ by clustering $\mathcal{P}_o$ using the mean shift [28] algorithm. Clustering approaches generally outperform direct end-to-end segmentation in point clouds [216], motivating this design choice.

7.3.2 Fruit Descriptor Extraction Module

Each fruit instance $\mathcal{F}_i$ is embedded into a fixed-size support region $\mathcal{S}_i$ to ensure local context:

$$\mathcal{S}_i = \{(\mathbf{p}, \mathbf{k}) \in \mathcal{P} \mid \|\mathbf{p} - \mathbf{c}_i\|_2 \leq s\}. \tag{7.2}$$

Unlike $\mathcal{F}_i$, which only contains the fruit's points, $\mathcal{S}_i$ captures neighboring structures, improving discriminability and robustness. The descriptor should remain invariant to changes in pose, shape, and color, while producing consistent embeddings across different acquisitions of the same fruit.

As shown in Figure 7.2, $\mathcal{S}_i$ is voxelized with size v and processed with a MinkowskiNet encoder [26]. The network leverages sparse 3D convolutions in a hierarchical structure: initial convolutions with batch normalization (BN) [71] and ReLU extract low-level features, then L residual blocks progressively downsample spatial dimensions while expanding feature channels. Each block combines main convolutional paths with residual or projection shortcuts.

The encoder output is aggregated via global average pooling, yielding the fruit descriptor $\mathbf{d}_i \in \mathbb{R}^z$. Collecting all descriptors we build a set of descriptors $\mathcal{D} = \mathbf{d}_1, \ldots, \mathbf{d}_{|\mathcal{F}|}$.

7.3.3 Fruit Descriptors Matching Module

The fruit descriptors matching module operates on sets of descriptors of fruits that belong to different time steps. Let $\mathcal{D}^t$ be the set of descriptors of all the fruits segmented at time t, i.e., $\mathcal{F}^t = \{\mathcal{F}_1^t, \ldots, \mathcal{F}_A^t\}$, and $\mathcal{D}^{t-1}$ the set of descriptors of all the fruits segmented at time $t-1$, i.e., $\mathcal{F}^{t-1} = \{\mathcal{F}_1^{t-1}, \ldots, \mathcal{F}_B^{t-1}\}$.

We want to match descriptors in $\mathcal{D}^t$ with the corresponding descriptors in $\mathcal{D}^{t-1}$. Moreover, we also want to label new or simply not visible fruits descriptors with a no-match label, $\varnothing$. Formally, we want to compute the association vector $\mathbf{y} = [y_1, \ldots, y_A]^\top$ with $y_i \in \{\varnothing, 1, \ldots, B\}$. Figure 7.2 below depicts the matching module's architecture.

For a query fruit $\mathcal{F}_q^t \in \mathcal{F}^t$ with descriptor $\mathbf{d}_q^t$, we compare it against $\mathcal{D}^{t-1}$. The descriptors are concatenated into a candidate set $\mathbf{G} \in \mathbb{R}^{B \times 2z}$, augmented with positional encodings $\mathbf{PE}(\mathbf{T})$ of the relative positions $\mathbf{T}$ obtaining $\mathbf{G}_e$. Formally:

$$\mathbf{G}_e = \mathbf{G} + \mathbf{PE}(\mathbf{T}). \tag{7.3}$$

Then, $\mathbf{G}_e$ is extended with a zero-vector row to represent the no-match case, yielding $\mathbf{G}_p \in \mathbb{R}^{(B+1) \times 2z}$. We compute $\mathbf{PE} : \mathbb{R}^3 \mapsto \mathbb{R}^z$ as a function of the relative position matrix $\mathbf{T}$ applied to each row as done in [128], i.e., we apply it to each individual coordinate.

$\mathbf{G}_p$ is processed with a linear projection, BN, leaky ReLU, and a transformer encoder [190]. A final linear layer outputs logits $\mathbf{p}_l \in \mathbb{R}^{B+1}$, normalized via softmax to $\mathbf{p}_n$. To enforce geometric plausibility, fruits more than h meters from the query are masked out. The predicted matching fruit's index y_q is given by:

$$y_q = \begin{cases} i, & \text{if } i < B+1 \\ \varnothing & \text{if } i = B+1 \end{cases}, \text{ with } i = \underset{i}{\operatorname{argmax}}\,(\mathbf{p}). \tag{7.4}$$

By predicting y_i for each fruit in $\mathcal{F}_i^t$, we can associate each fruit in $\mathcal{F}_i^t$ to fruits in $\mathcal{F}_i^{t-1}$ with the vector $\mathbf{y}$.

7.3.4 Batch Matching

When matching batches of fruits, we build the matrix $\mathbf{H}^* \in \mathbb{R}^{A\times(B+1)}$, where each row encodes the match probabilities of a query fruit. To avoid conflicting assignments, we apply a greedy matching algorithm: the highest-confidence pairs are matched first, after which matched fruits are removed from the candidate pool. This process continues until all queries are matched or assigned $\varnothing$.

7.3.5 Loss Function

We trained the fruit instance segmentation method following [122], using a loss function $\mathcal{L}_{\text{ins}}$ composed of two terms:

$$\mathcal{L}_{\text{ins}} = \mathcal{L}_{\text{sem}} + \lambda_{\text{off}}\mathcal{L}_{\text{off}}. \tag{7.5}$$

$\mathcal{L}_{\text{sem}}$ accounts for the fruit semantic segmentation, and is a weighted sum of a cross-entropy loss $\mathcal{L}_{\text{ce}}$ and a Lovász-Softmax loss [8] $\mathcal{L}_{\text{Lovász}}$:

$$\mathcal{L}_{\text{sem}} = \lambda_{\text{ce}}\mathcal{L}_{\text{ce}} + \lambda_{\text{Lov}}\mathcal{L}_{\text{Lovász}}. \tag{7.6}$$

$\mathcal{L}_{\text{off}}$, measures the discrepancy between the predicted and the ground truth offsets:

$$\mathcal{L}_{\text{off}} = \frac{1}{|\mathcal{P}_f|}\sum_{j=1}^{|\mathcal{P}_f|}\|\mathbf{o}_j - \hat{\mathbf{o}}_j\|_1, \tag{7.7}$$

where $\mathbf{o}_j \in \mathbb{R}^3$ is the j-th predicted offset and $\hat{\mathbf{o}}_j \in \mathbb{R}^3$ is the corresponding ground truth offset.

We trained our descriptor extraction and re-identification method end-to-end by batch matching with a weighted loss function, $\mathcal{L}_{\text{m}}$, composed of two terms:

$$\mathcal{L}_{\text{m}} = \mathcal{L}_{\text{ce}} + \lambda_{\text{inj}}\mathcal{L}_{\text{inj}}. \tag{7.8}$$

$\mathcal{L}_{\text{ce}}$ is the cross-entropy loss between the ground truth matrix $\hat{\mathbf{H}}$ and the predicted matrix $\mathbf{H}^*$:

$$\mathcal{L}_{\text{ce}} = \sum_{i=1}^{A}\sum_{j=1}^{B+1} -\hat{\mathbf{H}}_{ij}\log(\mathbf{H^*}_{ij}). \tag{7.9}$$

$\mathcal{L}_{\text{inj}}$ forces the network to learn a bijective function, i.e. the network should map distinct fruits of $\mathcal{F}^t$ to distinct fruits of $\mathcal{F}^{t-1}$ and vice versa, avoiding assigning multiple fruits from a particular set to the same fruit in the other set. $\mathcal{L}_{\text{inj}}$ ignores the no-match fruit and it is weighted by λ_{inj} in the final L_{m} loss computation. We compute $\mathcal{L}_{\text{inj}}$ as follows:

$$\mathcal{L}_{\text{inj}} = \lambda_{\text{inj}}(\mathcal{L}_{\text{row}} + \mathcal{L}_{\text{col}}), \tag{7.10}$$

where

$$\mathcal{L}_{\text{row}} = \sum_{i=1}^{A}\left|(\sum_{j=1}^{B}\mathbf{H}*_{ij}) - 1\right| \tag{7.11}$$

and

$$\mathcal{L}_{\text{col}} = \sum_{j=1}^{B}\left|(\sum_{i=1}^{A}\mathbf{H}*_{ij}) - 1\right|. \tag{7.12}$$

Timepoint	Point Cloud	Annotations	# Annotations	Matched With	# Unmatched
$t = 1$	$\mathcal{P}^1$	$\hat{\mathcal{F}}^1$	616	–	–
$t = 2$	$\mathcal{P}^2$	$\hat{\mathcal{F}}^2$	556	$\hat{\mathcal{F}}^1$	56
$t = 3$	$\mathcal{P}^3$	$\hat{\mathcal{F}}^3$	159	$\hat{\mathcal{F}}^2$	5

TABLE 7.1: Overview of the strawberry dataset. Each point cloud $\mathcal{P}^t$ corresponds to a different acquisition time. Ground truth fruit annotations are denoted as $\hat{\mathcal{F}}^t$.

7.4 Experimental Evaluation

7.4.1 Dataset

To evaluate the performance of our instance segmentation method, we conduct experiments on two datasets. The first, introduced in [152], hereafter referred to as the strawberry dataset, serves as our primary benchmark. It is used for both instance segmentation and the downstream re-identification task, as it provides temporally coherent annotations.

For more details about the datasets acquisition, please refer to their original works (Gené-Mola et al. [49] and Riccardi et al. [152]). The data were collected in a commercial greenhouse using a static, high-precision Faro Focus3D-X130 laser scanner (see Section 2.1.4), capturing the same row of strawberry plants at three different points in time, separated by approximately one week. We denote the point clouds as $\mathcal{P}^1$, $\mathcal{P}^2$, and $\mathcal{P}^3$, which contain ground-truth fruit annotations $\hat{\mathcal{F}}^1$, $\hat{\mathcal{F}}^2$, and $\hat{\mathcal{F}}^3$ with 616, 556, and 159 strawberries, respectively. Fruits in $\hat{\mathcal{F}}^2$ are associated to fruit in $\hat{\mathcal{F}}^1$. Let $\hat{\mathbf{y}}_{2,1}$ be the ground truth vector of associations. Similarly, fruits in $\hat{\mathcal{F}}^3$ are associated to fruits in $\hat{\mathcal{F}}^2$ with the vector of associations $\hat{\mathbf{y}}_{3,2}$. In $\hat{\mathbf{y}}_{2,1}$, 56 strawberries from $\hat{\mathcal{F}}^2$ are not matched with those in $\hat{\mathcal{F}}^1$. In $\hat{\mathbf{y}}_{3,2}$, 5 strawberries from $\hat{\mathcal{F}}^3$ are not matched with $\hat{\mathcal{F}}^2$. See details in Table 7.1.

The second, PFuji-Size [49], is a publicly available dataset designed for fruit detection and sizing in agricultural settings. It comprises high-resolution RGB images and photogrammetry-derived 3D point clouds of Fuji apple trees, captured under field conditions. Each scene is annotated with ground truth fruit instance description (center and radius). It is divided in two main acquisitions, the first captured in 2018 and the second captured in 2020, both featuring three Fuji trees. We used the 2018 acquisition for training and validation, while the 2020 acquisition for testing. While manually inspecting the dataset annotations, we noticed some fruits missing in the first two trees of the 2020 collection. For this reason, we only used the third tree's fruit annotations for testing.

7.4.2 Metrics

To evaluate the instance segmentation capabilities we use as metrics panoptic quality (PQ) [85], consisting of segmentation quality (SQ) and recognition quality (RQ), and intersection over union (IoU). IoU is defined by,

$$\text{IoU} = \frac{1}{|TP|} \sum_{(p,g)\in TP} \frac{|p \cap g|}{|p \cup g|}, \tag{7.13}$$

where p is the set of points belonging to a predicted instance, g is the set of points belonging to a ground truth instance and TP is the set of matched predicted and ground truth instance pairs (with $\frac{|p\cap g|}{|p\cup g|} \geq$ threshold, we set it to 0.5 as commonly done).

The metrics PQ, SQ and RQ are defined as

$$\mathrm{PQ} = \underbrace{\frac{\sum_{(p,g)\in TP_{\mathrm{seg}}} \mathrm{IoU}(p,g)}{|TP_{\mathrm{seg}}|}}_{\text{segmentation quality (SQ)}} \cdot \underbrace{\frac{|TP_{\mathrm{seg}}|}{|TP_{\mathrm{seg}}| + \frac{1}{2}|FP_{\mathrm{seg}}| + \frac{1}{2}|FN_{\mathrm{seg}}|}}_{\text{recognition quality (RQ)}}, \tag{7.14}$$

where true positives (TP_{seg}), false positives (FP_{seg}), and false negatives (FN_{seg}) represents matched pairs of segments, unmatched predicted segments, and unmatched ground truth segments, respectively [84].

To evaluate the matching performance we first compute the following metrics: correct matching (CM, correctly matching two strawberries), mismatching (MM, correctly detecting a strawberry as a matching one but relating it with the wrong corresponding strawberry), false matching (FM, incorrectly labeling a no-matching strawberry with a strawberry), true negative (TN, correctly labeling a strawberry with the no-match label), and false negative (FN, incorrectly labeling a matching strawberry with the no-match label). Based on these, we calculate the following scores:

$$\mathrm{F1_p} = \frac{2\mathrm{CM}}{2\mathrm{CM} + \mathrm{MM} + \mathrm{FM} + \mathrm{FN}}, \tag{7.15}$$

$$\mathrm{F1_n} = \frac{2\mathrm{TN}}{2\mathrm{TN} + \mathrm{FM} + \mathrm{FN}}, \tag{7.16}$$

$$\mathrm{mF1} = \frac{\mathrm{F1_p} + \mathrm{F1_n}}{2}, \tag{7.17}$$

The standard F1 score measures the goodness of the predictive capacity of a model, but it only focuses on the positive class. By averaging the $\mathrm{F1_p}$ and $\mathrm{F1_n}$ scores in (7.17), we get an indicator for both, mF1.

In our experiments, we trained the re-identification module to maximize the mF1 score in the validation set, in order to balance performance between correct matching and correct no-match predictions.

7.4.3 Implementation Details

We trained instance segmentation and re-identification separately, and in particular we only used the strawberry ground truth fruit annotations for the re-identification task. This enables a clear separation between the two modules allowing for easy integration and replacement of the segmentation component with alternative solutions.

We trained from scratch both instance segmentation methods, MinkPanoptic (ours) and superpoint transformer, on both datasets.

We trained MinkPanoptic with initial learning rate of 0.01 in strawberry, and 0.03 in apple dataset. In both, it was linealy decreased, at each epoch, with a decay coefficient of 0.97 In both datasets we trained superpoint transformer using the default learning rate parameters, i.e., an initial value of 0.01 with a cosine annealing scheduler with warmup.

We used the same, standard data augmentation techniques for both methods: random yaw rotation, X or Y axis flip, X or Y scale change (with a uniformly random scaling factor in the range [0.97, 1.03] for strawberry and [0.95, 1.05] for apples), and point jittering (with variance 0.03 m for strawberry and 0.1 m for apples).

For MinkPanoptic we optimized the bandwidth values on the validation set, finding the best value to be of 0.01125 m for the strawberry dataset and 0.035 m for the apple dataset (see Section 7.5.1).

dataset	method	class	IoU	RQ	SQ	PQ
strawberry	superpoint transformer	background	97.1	**100**	97.1	97.1
		strawberry	48.3	48.8	79.2	38.7
		average	72.7	74.4	88.1	67.9
	MinkPanoptic (ours)	background	98.9	**100**	98.9	98.9
		strawberry	80.9	75.8	84.3	63.8
		average	89.9	87.9	91.6	81.4
PFuji-Size	superpoint transformer	background	92.5	**100**	92.2	92.2
		apple	39.4	49.2	71.0	34.9
		average	65.9	74.6	81.6	63.6
	MinkPanoptic (ours)	background	94.6	**100**	96.6	96.6
		apple	50.1	52.1	78.3	40.7
		average	72.3	76.0	87.4	68.7
average	superpoint transformer	background	94.8	**100**	94.7	94.7
		fruit	43.9	49	75.1	36.8
		average	69.3	74.5	84.9	65.8
	MinkPanoptic (ours)	background	**96.8**	**100**	**97.8**	**97.8**
		fruit	**65.5**	**64.0**	**81.3**	**52.3**
		average	**81.1**	**82.0**	**89.5**	**75.1**

TABLE 7.2: Comparison of superpoint transformer [154] and our MinkPanoptic on the instance segmentation task. All values are in %.

Our descriptor extraction and matching module is trained from scratch, with a fixed learning rate of $3 \cdot 10^{-4}$. Input fruit point clouds were augmented with random rotation in the range [-30°, 30°] on each of the three axis, point jittering with variance $7 \cdot 10^{-4}$ m, and color jittering with variance 0.05. In the loss function, we used weights $\lambda_{\text{ce}} = 2$, $\lambda_{\text{Lov}} = 10$, $\lambda_{\text{off}} = 10$ and $\lambda_{\text{inj}} = 0.08$. We set support radius $s = 0.2$ m and voxel size $v = 5 \cdot 10^{-4}$ m for the fruit descriptor extraction module and maximum matching distance $h = 0.05$ m for the matching module based on empirical evaluation. We used $L = 4$ layers using hidden dimensions 8, 8, 16, 16, and 64 (8 is the number of channels produced by the convolution of the initial feature extraction, while the subsequent are each layer's produced number of channels). We set the matching module transformer layer's input channel dimension l to 512, feedforward dimension to 1024, and number of heads to 8.

To train the fruit descriptor extraction and re-identification model, we manually divided the training set $\left(\hat{\mathcal{F}}^1, \hat{\mathcal{F}}^2, \text{and } \hat{\mathbf{y}}_{2,1}\right)$ into two non-overlapping sets, grouping corresponding fruits based on their 3D position. We used the first group as training set, containing approximately 80 % of the strawberries, and the second as validation set.

7.4.4 Instance Segmentation Results

We compared our instance segmentation method, MinkPanoptic, with multiple baselines and on two fruit point cloud datasets. Surprisingly, state-of-the-art instance segmentation methods struggle when trained with small or very small data, providing poor results in a seemingly simple task (see Section 7.4.5). Only superpoint transformer [154] was able to accurately segment strawberries or apples instances, and for this reason we only report comparison with this baseline.

In the strawberry dataset, we trained on the first two point clouds, i.e., $\mathcal{P}^1$ as actual training set and $\mathcal{P}^2$ as validation set, and tested on the third point cloud, i.e., $\mathcal{P}^3$.

Due to the pre-processing requirements of the superpoint transformer, we constructed a dataset by extracting 800 annotated point clouds for training and 200 for validation from the three original point clouds. Each sample was generated by cropping around a randomly selected seed point within the strawberry row, resulting in segments 0.15 m wide, with a voxel size of 0.001 m. In contrast, MinkPanoptic was trained using the same number of point clouds, but with a larger crop width of 0.3 m and a finer voxel size of 0.0005 m, made possible by its lower memory usage, particularly during training.

The results in Table 7.2 show that MinkPanoptic achieves the best performance across both datasets, with clear gains on the strawberry data where the fruit's spherical geometry matches well with its offset prediction and mean-shift clustering. The method also demonstrates superior performance on the apple dataset, confirming robustness to different fruit types. All methods reach 100% recognition of background points, but MinkPanoptic yields the largest improvements in panoptic quality and IoU for the fruit class, demonstrating its effectiveness for precise segmentation in agricultural settings.

7.4.5 Discussion: Instance Segmentation state-of-the-art performance

Surprisingly, current state-of-the-art instance segmentation methods (i.e., [218, 161, 122, 89, 167]) struggle when trained with small or very small data, providing poor results in a seemingly simple task, such as the segmentation of strawberries or apples. There are many reasons why. Most methods are best suited for LiDAR point clouds, e.g. P3Former [218], exploiting their cylindrical space distribution. Methods like Mask3D [161], MaskPLS [122], OneFormer3D [89], and Spherical Mask [167] suffer in this task due to their reliance on voxelization, which can lead to the loss of small object details during downsampling. Using small voxels to preserve these details dramatically increases computational costs, making it impractical for our application, while increasing the voxel size to lower the computational complexity results in missing the small objects. Moreover, the multi-resolution technique of these methods, usually helpful, deteriorates the performance on dense, small-scale datasets like strawberries, and the learned queries might collapse to similar representations, failing to differentiate between individual fruits. We made every effort to include these state-of-the-art methods in our comparison; however, due to the severe limitations described above, their performance was so poor that they failed to produce even basic, usable segmentation results on our datasets.

7.4.6 Re-identification Results

In this experiment, we evaluate fruit re-identification using the strawberry dataset, where predicted instances in $\mathcal{P}^3$ are matched against ground truth annotations of $\mathcal{P}^2$. Evaluation is performed at IoU thresholds between 5% and 30%, balancing the tradeoff between allowing noisy predictions and discarding false positives. Each ground truth fruit is assigned the ID of the predicted instance with maximum IoU above threshold, while unmatched predictions are assigned new IDs.

As baselines, we consider a nearest neighbor approach with an Optuna-optimized distance threshold ($\epsilon^* = 0.033$, m) and Riccardi et al. [152], which relies on histogram descriptors of local fruit distributions. Nearest neighbor captures spatial proximity but struggles with unmatched cases, while Riccardi's method encodes relative geometry. In our experiments, we use the parameter setting suggested in the original implementation. We compare only with these two baselines because, to the best of our knowledge, Riccardi et al. is the

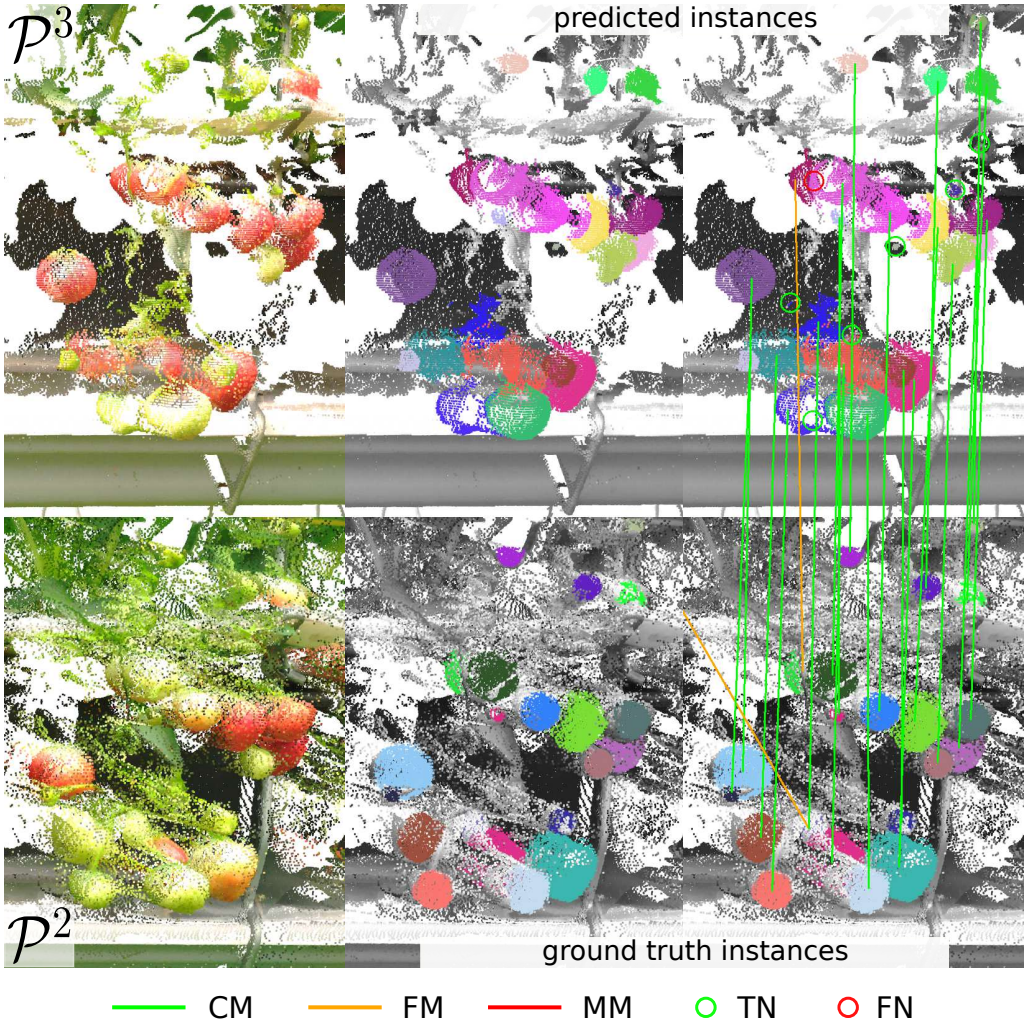


FIGURE 7.3: Fruit instance segmentation and re-identification using our method, in a good performance scenario. On the top are point clouds from $\mathcal{P}^3$ that we match with the ground truth fruit annotations on $\mathcal{P}^2$ depicted below. From left to right: the original, colored point clouds; the instance segmentation (ground truth for $\mathcal{P}^2$, predicted using MinkPanoptic for $\mathcal{P}^3$); the re-identification results. Green lines indicate correct matches (CMs), red lines indicate false matches (FMs), orange lines indicate mismatches (MMs), green circles indicate true no-matches (TNs) and red circles indicate false no-matches (FNs). Most matches are correct, with only two FMs and one FN. Best viewed in color.

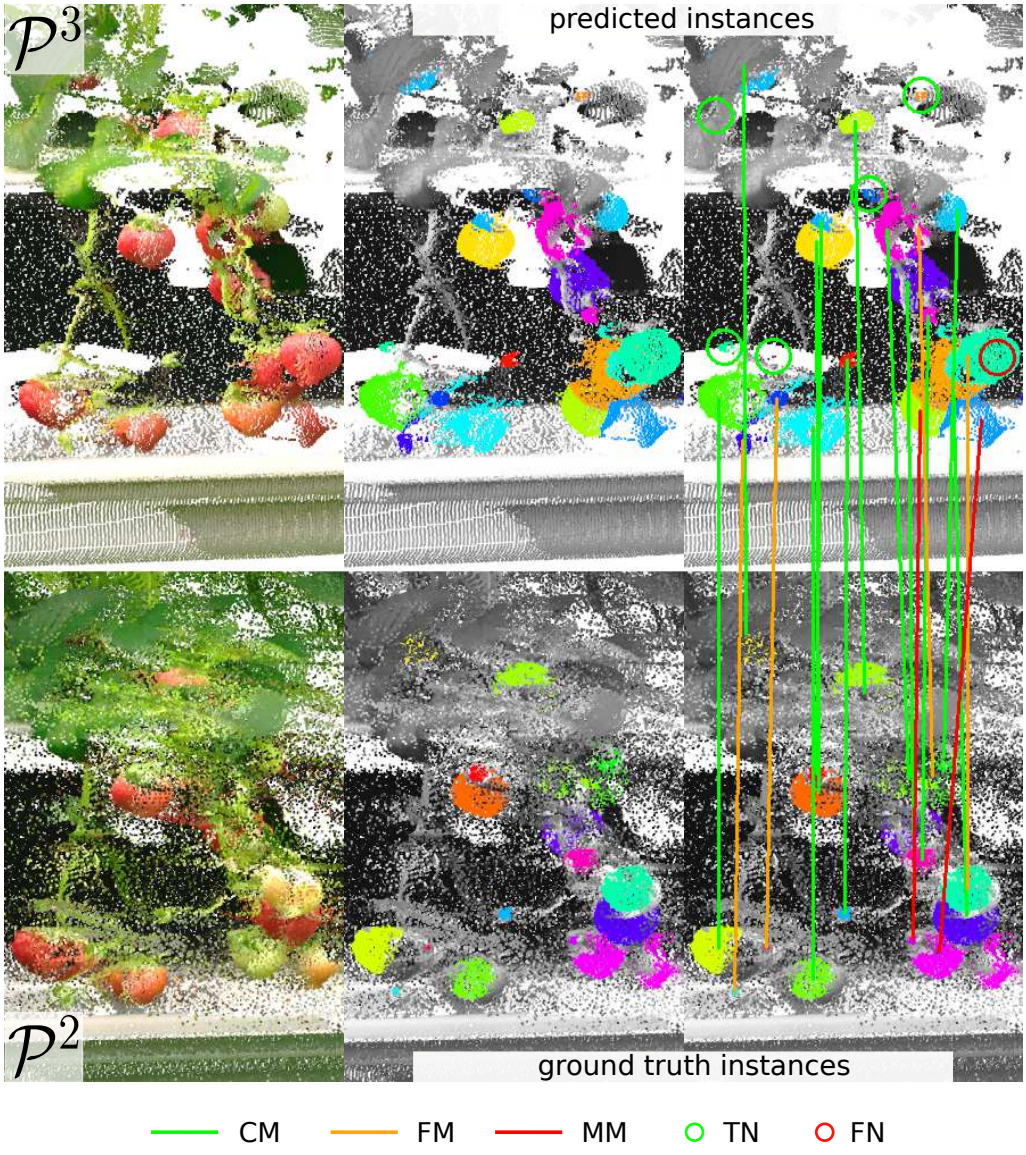


FIGURE 7.4: Fruit instance segmentation and re-identification using our method, in a difficult scenario. On the top are point clouds from $\mathcal{P}^3$ that we match with the ground truth fruit annotations on $\mathcal{P}^2$ depicted below. From left to right: the original, colored point clouds; the instance segmentation (ground truth for $\mathcal{P}^2$, predicted using MinkPanoptic for $\mathcal{P}^3$); the re-identification results. Green lines indicate correct matches (CMs), red lines indicate false matches (FMs), orange lines indicate mismatches (MMs), green circles indicate true no-matches (TNs) and red circles indicate false no-matches (FNs). Many matches are correct, but there are also several FMs, MMs and one FN. Best viewed in color.

IoU level	using MinkPanoptic prediction									using SPT prediction								
	NN			Riccardi et al.			Ours			NN			Riccardi et al.			Ours		
	$F1_p$	$F1_n$	**mF1**	$F1_p$	$F1_n$	**mF1**	$F1_p$	$F1_n$	**mF1**	$F1_p$	$F1_n$	**mF1**	$F1_p$	$F1_n$	**mF1**	$F1_p$	$F1_n$	**mF1**
GT	84.7	**75.0**	**79.9**	**92.6**	0.0	46.3	76.4	51.4	63.9	84.7	**75.0**	**79.9**	**92.6**	0.0	46.3	76.4	51.4	63.9
@5%	76.3	38.9	57.6	73.5	38.3	55.9	**81.5**	**63.9**	**72.7**	**80.8**	21.4	51.1	65.8	9.1	37.5	80.6	**50.0**	**65.3**
@10%	74.9	35.4	55.1	71.5	35.6	53.6	**80.0**	**59.6**	**69.8**	**78.0**	15.4	46.7	69.9	6.1	38.0	76.2	**39.2**	**57.7**
@15%	73.3	33.3	53.3	70.1	35.8	53.0	**78.1**	**56.9**	**67.5**	**74.1**	11.5	42.8	60.5	7.0	33.8	71.2	**34.4**	**52.8**
@20%	71.8	30.4	51.1	68.8	36.8	52.8	**75.7**	**53.0**	**64.4**	**67.2**	8.7	38.0	55.0	8.1	31.6	64.9	**27.2**	**46.0**
@25%	69.0	28.0	48.5	66.2	34.4	50.3	**73.5**	**51.2**	**62.4**	**59.6**	7.1	33.4	49.8	9.0	29.4	57.8	**22.9**	**40.4**
@30%	61.1	22.8	41.9	60.6	34.5	47.5	**65.7**	**44.6**	**55.2**	**51.6**	5.9	28.8	38.1	7.5	22.8	50.3	**21.2**	**35.7**
avg	73.0	37.7	55.3	71.9	30.8	51.3	**75.8**	**54.4**	**65.1**	**70.9**	20.7	45.8	61.7	6.7	34.2	68.2	**35.2**	**51.7**

TABLE 7.3: Re-identification performance across various IoU thresholds using different instance segmentation methods (MinkPanoptic (ours) and superpoint transformer [154]) and matching baselines (Nearest Neighbor (NN), Riccardi et al. [152] and ours).

only publicly available work addressing fruit re-identification in point clouds, while the nearest neighbor approach is a simple yet effective baseline. Also, Riccardi et al. demonstrated superior capabilities compared to other traditional descriptor-based matchings.

Results in Table 7.3 and Figure 7.5 show that, with MinkPanoptic predictions, our method consistently achieves the highest test performance across all metrics, reaching an average mF1 of 65.1%, a 9.8% improvement over the second-best. Nearest neighbor surpasses Riccardi in most settings but remains inferior to our approach. With superpoint transformer predictions, our method also yields the strongest results, especially for $F1_n$ and mF1, achieving 51.7% mF1, 5.9% above the next best. While nearest neighbor obtains the best $F1_p$ due to reliance on spatial closeness, our method balances positives and negatives more effectively, particularly under noisy segmentations.

In Figure 7.3 and Figure 7.4, we show qualitative results of our re-identification method using the instance segmentation predicted by MinkPanoptic. In Figure 7.3, the re-identification is almost perfect, with only two FMs and one FN. In Figure 7.4, the re-identification is less accurate, with several FMs, MMs, and one FN. This is mainly due to the instance segmentation errors, such as merged fruits (leading to the FNs on bottom right) or false detections (leading to the FNs on bottom left). Nonetheless, most detection faults are correctly handled by our re-identification method with several TNs.

Figure 7.5 quantitatively highlights our approach's superior performance. The radar plot of our method has always a greater area with respect to the baselines when using MinkPanoptic's predictions. With SPT's, the area is slightly smaller only on the $F1_p$ radar plot, while the area is sharply larger on the $F1_n$ and mF1 radar plots.

For a broader discussion of the results, the method's limitations and future directions, refer to Section 7.5.3.

7.5 Ablation Study

7.5.1 Instance Segmentation

We fine-tuned the mean shift bandwidth parameter for MinkPanoptic using the validation sets of the strawberry and PFuji-Size datasets. The bandwidth optimization process is a hyperparameter that is not involved in the training process of MinkPanoptic, since the loss only cares for binary semantic segmentation and offset prediction. Thus, we were able to adjust it after the training by running multiple instance segmentations and analyzing the performance. By doing so in a grid-search approach, we could identify the optimal bandwidth value that

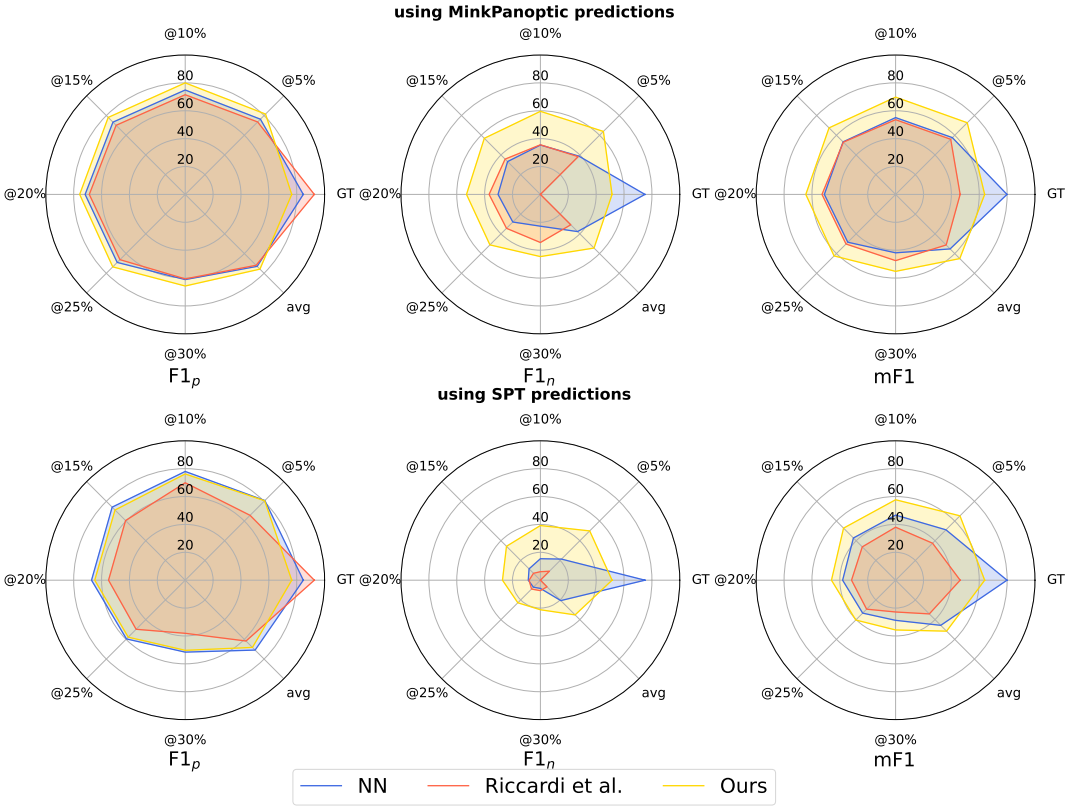


FIGURE 7.5: Radar plots comparing re-identification performance, detailed in Table 7.3, at varying IoU thresholds using instance segmentations from our method (MinkPanoptic) and SuperPoint. Best viewed in color.

maximizes the PQ on the validation set. Performance, measured by PQ, followed a near-parabolic trend, allowing clear selection of optimal bandwidth values reported in Table 7.4. Interestingly, the bandwidth corresponding to the average fruit radius did not yield the best performance: 0.094 m for strawberries and 0.07 m for apples were suboptimal, indicating that optimal clustering requires a smaller or larger radius than the mean fruit size.

7.5.2 Re-identification

We conducted experiments to evaluate the impact of various fruit descriptor extractor design choices on our method. We performed a 5-fold cross-validation on the strawberry training set $\left(\hat{\mathcal{F}}^1, \hat{\mathcal{F}}^2, \hat{\mathbf{y}}_{2,1}\right)$, dividing the data spatially. Each configuration was repeated four times with different seeds, averaging the results to reduce variability (Figure 7.7).

First, with a fixed matching module, varying encoder hidden dimensions showed that [8, 8, 16, 16, 64] achieved the highest performance, with mF1 $=$ 61.2%, F1_p=75.5%, and F1_n=47.0%. Increasing the transformer's feedforward dimension from 512 to 1024 provided no improvement, with the best alternative configuration reaching at most mF1 $=$ 60.2%.

Next, we tested incorporating neighboring fruits via graph-based descriptors, using GCNConv and EdgeConv operators. GCNConv with feedforward dimension 512 achieved mF1 $=$ 61.0% and the highest F1_n=47.1%, while EdgeConv with feedforward 1024 reached the highest F1_p=76.9% but reduced overall mF1 due to imbalance between positive and negative matches.

In conclusion, the graph-free descriptor extraction with encoder dimensions [8, 8, 16, 16, 64] and transformer feedforward 512 achieved the best overall performance in cross-validation, providing the most balanced and robust re-identification results.

strawberry				PFuji-Size			
bw (m)	**RQ**	**SQ**	**PQ**	**bw (m)**	**RQ**	**SQ**	**PQ**
0.0094†	80.2	**86.2**	71.5	0.02	80.2	**85.3**	70.4
0.0100	80.5	**86.2**	71.7	0.03	85.3	85.2	74.1
0.0105	80.8	86.1	71.9	0.0325	85.6	85.2	74.4
0.0110	80.8	86.1	71.9	**0.035**	85.8	85.2	**74.5**
0.01125	**81.0**	86.0	**72.0**	0.0375	85.7	85.1	74.3
0.0115	80.7	86.1	71.8	0.04	85.7	85.1	74.3
0.01175	80.8	86.1	71.9	0.0425	**86.0**	85.0	74.4
0.1200	80.8	86.1	71.9	0.07†	81.9	84.6	71.2
0.1300	80.3	86.1	71.5	0.09	74.6	84.5	65.7

TABLE 7.4: Effect of mean shift bandwidth fine-tuning on the instance segmentation performance on the strawberry and PFuji-Size datasets. Metrics are on the valid set in %. † symbol indicates the average radius size of the strawberries and apples in the valid set.

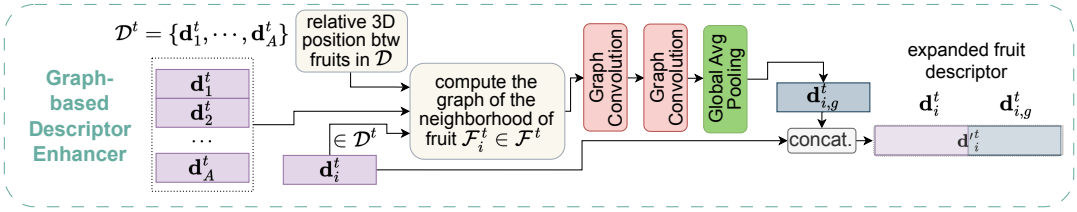


FIGURE 7.6: We investigated the significance of neighboring fruits in the descriptor computation. The module builds the graph of the neighborhood of fruit $\mathcal{F}_i^t$. Two graph convolutions process the graph and a final global average pooling computes the neighborhood descriptor $\mathbf{d}_{i,g}^t$, which is concatenated to the original $\mathcal{F}_i$ fruit descriptor $\mathbf{d}_i^t$.

7.5.3 Discussion of the Results, Limitations and Future Works

Our findings demonstrate the potential of leveraging high-resolution 3D point clouds for fruit monitoring, particularly in agricultural environments. The proposed method effectively combines instance segmentation and re-identification, achieving state-of-the-art performance on challenging datasets. However, several aspects need further discussion.

First, the results highlight the importance of accurate instance segmentation as a precursor to successful re-identification. While our method outperformed baselines, its performance is inherently tied to the quality of the instance segmentation. Errors such as merged fruits or false detections can propagate to the re-identification stage, reducing overall accuracy. This highlights the need for robust segmentation methods tailored to agricultural datasets, which often feature dense, occluded, and visually similar objects, unlike indoor or urban datasets.

Second, the experiments reveal the adaptability of our approach to different fruit types and datasets. The use of MinkPanoptic for instance segmentation proved effective across both strawberry and apple datasets, suggesting its generalizability. However, the re-identification module was only evaluated on the strawberry dataset due to the lack of publicly available temporally consistent instance-annotated datasets for other fruits. Expanding the evaluation to diverse datasets and fruit types would provide a more comprehensive understanding of the method's robustness and limitations.

Third, the ablation study highlights the significance of design choices in the descriptor extraction and matching modules. While incorporating graph-based features showed potential, the simpler graph-free approach achieved the best balance between positive and negative

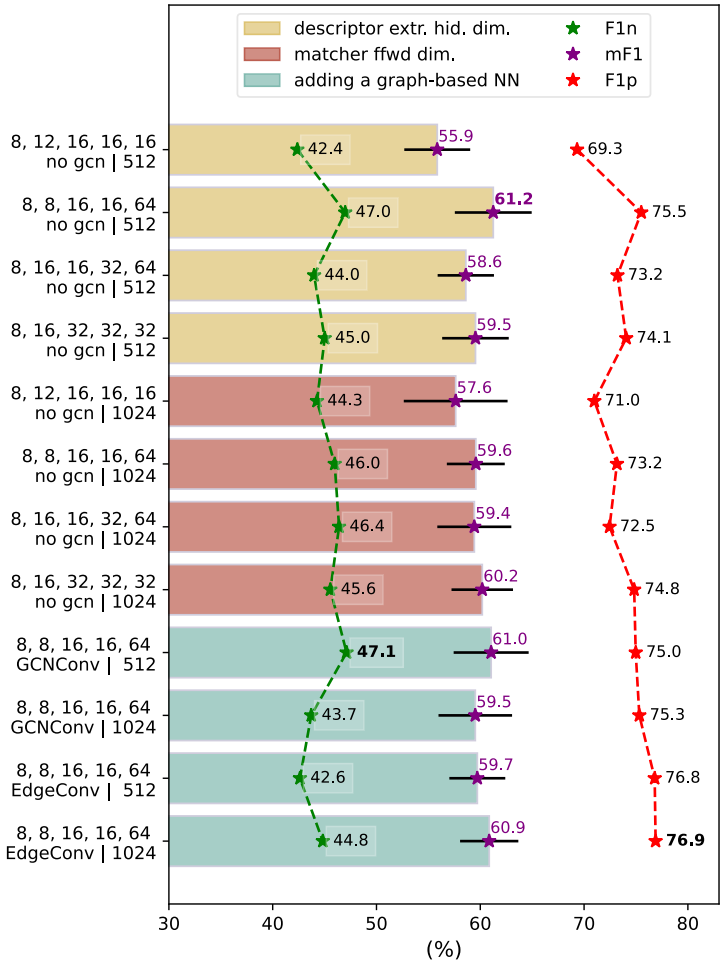


FIGURE 7.7: Ablation study to evaluate the impact of various design choices on our method. All values are in %. The horizontal lines depict the standard deviation of mF1. The first four bars colored in refer to the graph-free method with different fruit descriptor encoder hidden dimensions. The following four bars colored in refer to the graph-free method with different transformer feedforward dimensions. The last four bars colored in refer to the graph-based method with different graph convolutional operators and transformer feedforward dimensions. The best performing method for each of the $F1_n$, $F1_p$ and mF1 metrics is highlighted in bold. The mF1 metric is the most important, as it balances positive and negative predictions, and is used to determine the overall best performing method.

predictions. This suggests that the added complexity of graph-based methods may not always translate to improved performance, particularly in scenarios with limited training data.

Finally, the broader implications of this work extend beyond fruit monitoring. The proposed method can be adapted to other domains requiring instance segmentation and temporal matching in 3D point clouds, such as forestry, construction, and urban planning. However, the reliance on high-resolution LiDAR data may limit its applicability in resource-constrained settings. Future work could explore the use of lower-cost sensors or hybrid approaches combining 2D and 3D data to enhance accessibility.

7.6 Conclusion

In this chapter, we presented a novel approach for temporal fruit monitoring based on fruit instance segmentation and re-identification on colored point clouds acquired by a high-resolution LiDAR scanner at different points in time. Our method first segments fruits using a learning-based instance segmentation approach, and then each segmented fruit is processed by a 3D sparse convolutional neural network to compute a compact descriptor. Each fruit is matched with its corresponding instance from a previous data collection by using their descriptors and attention-based matching network designed for robust temporal association. We implemented and evaluated our approach on real-world 3D datasets and provided comparisons to other existing techniques. The experiments supported all claims made in this article and demonstrated that our method achieves superior performance in both instance segmentation and re-identification, even when trained on ground truth annotations but tested on predicted instance segmentation. This flexibility makes our method highly adaptable to different scenarios, highlighting its potential for broader applications in 3D pattern recognition tasks where segmentation and temporal consistency are critical, allowing for easy integration and replacement of the segmentation component with alternative solutions.

Chapter 8

Real-time Underwater Place Recognition in Synthetic and Real Environments using Multibeam Sonar and Learning-based Descriptors

AUTONOMOUS underwater navigation presents a unique set of challenges, the main one being reliable localization. Traditional positioning systems such as GPS or USBL are often unstable or noisy in underwater environments, limiting their effectiveness. A key solution to this problem is place recognition, but in underwater scenarios cameras are often subject to water turbidity and low-light conditions, making their use unreliable. Sonars, by contrast, are robust to these conditions but produces more difficult to interpret data. In this chapter we propose a novel method for underwater place recognition, designing a global descriptor for multibeam sonar images that allows the AUV to compare its current observation with a database of descriptors obtained from known locations, even in sparsely structured environments.

8.1 Motivation

Autonomous underwater vehicles (AUVs) are increasingly used for tasks such as environmental monitoring, infrastructure inspection, and search-and-rescue operations. However, their effectiveness is limited by the difficulty of underwater localization. Unlike surface vehicles that rely on stable GPS signals, AUVs operate in environments where standard positioning systems, such as USBL, are noisy, unstable, or require external infrastructure, making robust self-localization a critical challenge.

Underwater visual place recognition (VPR) addresses this challenge. [160] defines the VPR task as involving the matching of one or multiple image sets in order to determine which images show the same places in the world. Essentially, VPR is an image retrieval problem where the context is to recognize previously seen places. Optical cameras are often unreliable in underwater conditions due to low light and turbidity [212], whereas sonar sensors are robust to these factors. In particular, multibeam sonar captures detailed spatial information but produces noisy images affected by speckle [228], complicating interpretation.

Handcrafted features that typically excel with optical images often doesn't succeed with sonar images, due to their inherent high noise and low resolution. To leverage sonar for VPR, we propose a deep learning approach. Deep learning neural networks are particularly suited, thanks to their noise removal capabilities achieved during training. Our deep learning

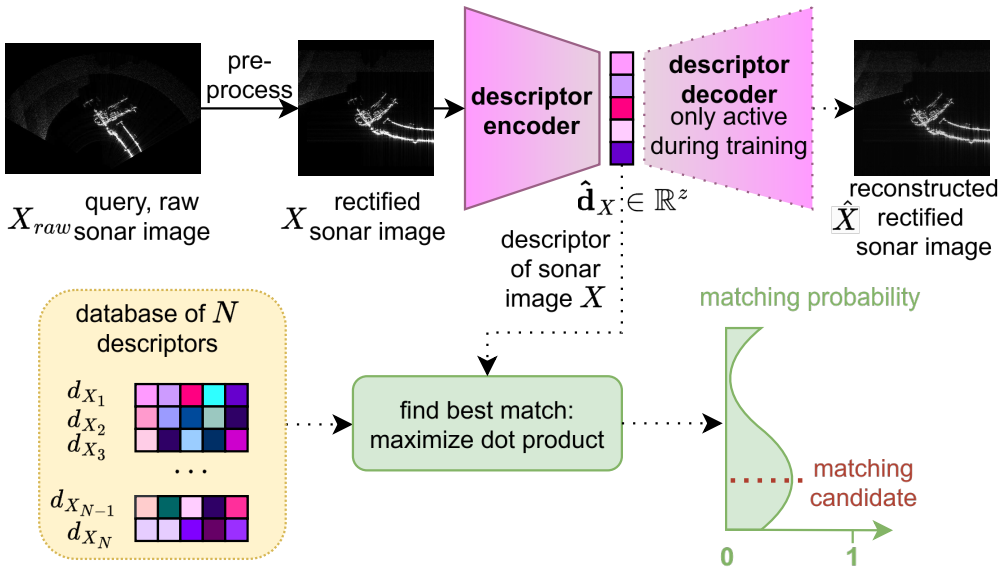


FIGURE 8.1: Overview of the proposed approach employing a pre-processing step for preparing the input sonar image, a feature encoder for computing the sonar image descriptor, and the matching strategy that searches a descriptor database to identify the best match.

neural network computes global descriptors for multibeam sonar images, and is optimized for solving two tasks: place recognition and sonar image reconstruction. The reconstruction task serves as an unsupervised proxy to improve feature learning, even with partially annotated data. Descriptors are trained so that Euclidean distances between them reflect the oriented-intersection over union (o-IoU) between the corresponding AUV poses, capturing spatial similarity. We further introduce a batch contrastive learning method, inspired by CLIP, that aligns the descriptor similarity matrix with the ground-truth dyadic kernel matrix for a batch of sonar images, computed using as kernel function the sonar image o-IoU. We focus on the x, y, and yaw dimensions, reflecting the typical AUV operations where roll and pitch are stabilized and altitude is precisely measured.

We evaluate the method in two settings: (i) training only on synthetic sonar image–pose pairs, and (ii) hybrid training where synthetic images are used for place recognition and reconstruction, while real images only for reconstruction. Testing on real sonar images in sparsely structured environments demonstrates that our approach enables efficient, unsupervised place recognition and reliable AUV localization while addressing domain adaptation challenges.

8.2 Related Work

8.2.1 AUV Localization

Underwater localization methods are generally based on measurements from beacons with stationary acoustic transponders deployed in the seabed as in [119, 229] or ultra-short baseline acoustic positioning (USBL) systems installed either on the AUV and on a support vessel or nearby buoys, as in [207]. The known landmark-based map of the beacons or buoys is used as the solution to a SLAM (Simultaneous Localization And Mapping) problem, such as that described in [138]. [101] comprises a recent review of techniques addressing the underwater SLAM problem. Alternatively, other approaches use natural landmarks, or salient features, extracted from sonar or optical images, such as those presented in [102, 149]. Typically,

most of the methods rely on probabilistic techniques such as particle filters, nonlinear versions of the Kalman filter (e.g., the extended Kalman filter (EKF) presented in [187]), or pose-graphs (described in [55]). A recent review of AUV localization techniques can be found in [124]. Other methods use measurements of seabed topography to localize the AUV relative to a given bathymetric map [158, 185, 227]. For example, [227] solves SLAM by fusing bathymetric information and acoustic signals collected by communicating with beacons which have unknown positions. Such acoustic signals can be used to produce range data associations. Similarly, the presence of human-made infrastructures having known positions can be used for the AUV localization [125, 81, 126]. In [125], the authors propose a sonar-based localization approach that uses a particle filter to match sonar data with a volumetric, partially unknown map of the underwater environment. Instead, [81] present a vision-based localization method for structured underwater environments, in which a template-based object detection technique is applied together with a Monte Carlo localization (MCL) algorithm. In [126] the authors propose a localization algorithm for AUVs moving within a port or naval pier. Localization is achieved by matching aerial images of infrastructures with synthetically generated overhead images from sonar images using a convolutional neural network. Our method falls into this last class of methods and, being object-detection-based, recalls the technique proposed in [81] but, compared to this, we rely on sonar data and exploit a deep learning-based orientation classification.

8.2.2 Sonar-based Underwater Visual Place Recognition

VPR from a query image typically involves leveraging a database of reference images and extracting distinguishing features for comparison. To achieve this, feature descriptors play a pivotal role and are mainly classified into either the local or global category.

Handcrafted local [117, 75] and global [136] features that typically excel with optical images often fall short in the complex conditions of sonar imaging. To address this, [103] introduced the use of CNN-learned features as descriptors, marking a shift towards deep learning solutions. Building on this idea, [14] proposed a siamese CNN architecture to compute similarity scores for sonar images. A notable advancement came from [151], which adapted PoseNet with a triplet loss, a technique popular in face recognition, and trained it on a synthetic sonar sensor achieving strong results on real sonar datasets. Similarly, [96] leveraged triplet loss with ResNet to capture spatial metrics in side-scan sonar data, while [121] utilized convolutional autoencoders and a fusion voting strategy to enable unsupervised learning of key underwater landmarks. Recently, [36] proposed a sonar image descriptor-based place recognition system where the descriptor is computed by a ResNet and a random Gaussian projection layer. The system is trained and tested only on synthetic datasets. These approaches collectively demonstrate the promise of deep learning in overcoming the limitations of traditional methods for sonar-based localization. Our approach focuses on place recognition leveraging multibeam sonar images and learning-based descriptors, employing a deep neural network trained with a novel training approach to optimize descriptor similarity based on oriented-IoU, enabling accurate localization through efficient descriptor matching.

8.3 Our Approach

Sonar images are initially captured in a fan-shaped format, reflecting the geometry of the sonar sensor beams which radiate outward from a central origin, with a defined horizontal field of view. We perform a pre-processing step that converts a raw sonar image X_{raw} into a rectified one X, where each beam of the sonar is aligned as a column and the rows correspond to the range values. The conversion is given by the formula:

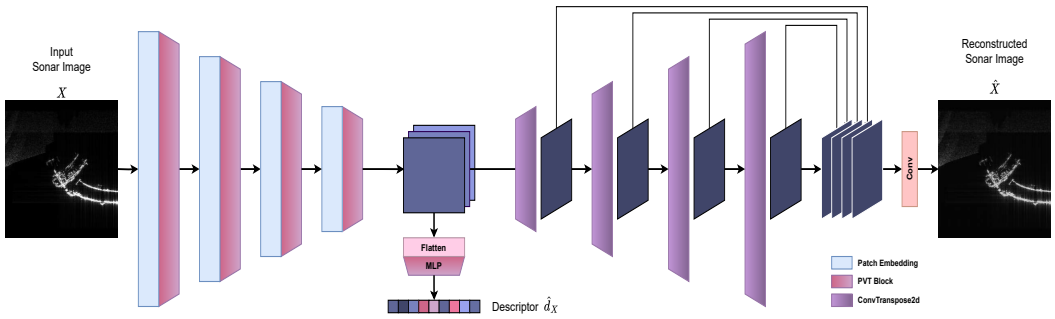


FIGURE 8.2: A representation of the proposed encoder-decoder network architecture for the underwater place recognition task. The encoder, shown on the left, consists of four stages, each integrating a patch embedding layer and pyramid vision transformer (PVT) [201] blocks. On the right, the decoder mirrors this structure with four stages of 2D transpose convolution layers, each processing feature maps from the corresponding encoder stage. The center highlights the description computation, utilizing the final feature map from the encoder.

$$X[n,b] = X_{raw}\left(\left\lfloor y_o - \frac{r_n \cdot \sin(\phi_b)}{p_{res}}\right\rfloor, \left\lfloor x_o + \frac{r_n \cdot \cos(\phi_b)}{p_{res}}\right\rfloor\right) \tag{8.1}$$

The beam angle for b-th beam is calculated as $\phi_b = \phi_{start} - b \cdot \Delta\phi$, where ϕ_{start} is the start and $\Delta\phi = \phi / B$ is the angular resolution based on number of beams B. Similarly, the range for the n-th bin is $r_n = n \cdot r_{res}$, where $r_{res} = R_{max}/N$ is the range resolution derived from the maximum range R_{max} and the number of range bins N. The origin (x_o, y_o) specifies the pixel coordinates of the center in X_{raw}, and p_{res} denotes the pixel resolution for mapping distances into the image grid.

Following this rectification, the sonar image is transformed into a two-channel tensor by applying cosine and sine functions to each pixel intensity. These channels are stacked together to form the input image of size $H \times W \times 2$.

8.3.1 Network Architecture

The proposed network (see Figure 8.2) follows an encoder-decoder design, combining the encoder from the Pyramid Vision Transformer (PVT) [201] for descriptor extraction and the decoder from SegFormer [220] for sonar image reconstruction. The dual objective is to extract a robust, discriminative descriptor while reconstructing the input image to enhance feature learning.

The PVT encoder processes sonar images hierarchically in four stages, generating multi-scale feature maps $[F_1, F_2, F_3, F_4]$ with scaling factors $[4, 8, 16, 32]$ and embedding dimensions $[16, 32, 64, 128]$. Each stage splits its input into patches, applies transformer blocks with spatial reduction attention [201], and produces down-sampled feature maps. This hierarchical design captures both coarse and fine-grained patterns, yielding rich representations. The final feature map F_4 is flattened and projected via an MLP to a 1024-dimensional L_2-normalized descriptor d_X for the input sonar image X.

The SegFormer-inspired decoder reconstructs the input sonar image $\hat{X}$ by up-sampling all encoder feature maps through transpose convolutions, aligning them in resolution and embedding dimension, then concatenating and refining them with a final convolution. This reconstruction acts as a self-supervised proxy task, improving the alignment and discriminative power of the extracted descriptor.

8.3.2 Synthetic Dataset Generation

The training and evaluation of deep learning models often requires to acquire and annotate large quantities of data. Collecting sonar images and annotating accurate AUV poses is especially expensive and time-consuming. For this reason, we generated a synthetic dataset of pairs of multibeam sonar images and poses using the simulation tool proposed in [234]. We simulated a Tritech Gemini 720i multibeam sonar sensor with 30 m range and 120° horizontal aperture. The simulated assets are designed to be consistent with assets in real-world. Specifically, we used simplified 3D CAD models of underwater structures. All the structures are placed at the center of the simulation, with two principal assets: a base structure positioned at 9.40 m depth, a flying structure positioned at 4.27 m depth, covering an area of approximately 12.2 m × 6.1 m. As in [36, 137], we generate a 2D grid around the target object, and for each valid pose in the grid, we acquire a simulated sonar image with the sensor oriented towards the object. The simulated sonar has a horizontal field of view of 360°, centered on the target asset, with zero roll and pitch, and all acquisitions are at the same altitude. Each image is paired with the pose of the AUV relative to the object.

8.3.3 Batch Contrastive Learning

We present a new training method called batch contrastive learning, designed to learn descriptors for place recognition. The goal is to ensure that the Euclidean distance between descriptors reflects the spatial proximity of the locations and orientations where the corresponding data was captured.

Let's define the sonar fan of a sonar image as the region of 2D space that corresponds to the circular sector of a circle, centered on the AUV's position and with a radius r, whose central angle's bisector coincides with the front-looking axis of the sonar. See Figure 8.3 for sonar fan examples.

We define the oriented-intersection over union (o-IoU) of two sonar fans F_1 and F_2, as:

$$\text{o-IoU}(F_1, F_2) = \frac{p \times intersection(F_1, F_2)}{union(F_1, F_2)} \tag{8.2}$$

It corresponds to the IoU multiplied by the scaling coefficient p, computed as:

$$p = \left| \cos\left(\frac{\gamma_1 - \gamma_2}{2} \right) \right| \tag{8.3}$$

where γ_1, γ_2 are the yaw orientations of poses of the two sonar images.

Now, consider a batch of rectified sonar images $\{X_1, ..., X_k\}$, their corresponding poses $\{p_1, ..., p_k\}$, predicted descriptors $\{\hat{d}_1, ..., \hat{d}_k\}$ and sonar fans $\{F_1, ..., F_k\}$. We compute two symmetric matrices, $\mathbf{D}, \mathbf{P} \in \mathbb{R}^{k \times k}$. The element $\mathbf{D}_{ij}$, at row i and column j, corresponds to the complement of the ground truth o-IoU of the two sonar fans F_i and F_j relative to the poses p_i and p_j, in other words $\mathbf{D}_{ij} = 1 - \text{o-IoU}(F_i, F_j)$. Instead, each $\mathbf{P}_{ij}$ is obtained by computing the Euclidean distance between the predicted descriptors $\hat{d}_i$ and $\hat{d}_j$, in other words $\mathbf{P}_{ij} = \|\hat{d}_i - \hat{d}_j\|_2$. The goal is to predict $\mathbf{P}$ as similar as possible to $\mathbf{D}$. This is obtained by minimizing the cross entropy loss $\mathcal{L}_{bcl}$ between the predicted and ground truth oriented-IoU matrices $\mathbf{P}$ and $\mathbf{D}$. At test time, we match the predicted descriptor of a query sonar image with a database of sonar images for which the position is known. By retrieving the descriptor for which the pairwise Euclidean distance is lower, we can compute the most similar pose in the database to the ground truth pose of the query sonar image.

training set		validation set (synthetic)			test set (real)		
synthetic	real*	pos error (m, ↓)	orie error (°, ↓)	o-IoU (%, ↑)	pos error (cm, ↓)	orie error (°, ↓)	o-IoU (%, ↑)
✓		**1.48**	**14.61**	**77.99**	**4.56**	**1.03**	**98.73**
✓	✓	**1.48**	16.02	77.61	9.46	2.92	97.36
random weights		1.81	19.34	73.13	5.15	1.78	98.60

TABLE 8.1: Results of our approach for underwater place localization, showing localization and o-IoU errors on validation and test sets under different training strategies. *: real data used in training are used only for reconstruction.

8.3.4 Training Strategy

We trained our network to fulfill two complementary tasks. The main one is a fully supervised VPR task (i.e., the ground truth poses of the AUV are required). We train only on synthetic sonar images minimizing the $\mathcal{L}_{bcl}$ loss (see Section 8.3.3). Then, as a proxy task, we enhance the sonar image feature extractor by minimizing the mean absolute error loss $\mathcal{L}_{mae}$ between predicted and ground truth sonar images $\hat{\mathbf{X}}$ and $\mathbf{X}$, combining also contributions from the last two decoder stages, weighted at 0.25 and 0.125, respectively.

The final loss function is the weighted sum of the two losses, with $\lambda = 10$, empirically shown to improve training stability and results:

$$\mathcal{L} = \lambda \mathcal{L}_{bcl} + \mathcal{L}_{mae}.$$

8.4 Experimental Evaluation

8.4.1 Dataset

As described in Section 8.3.2, we generated 6481 synthetic pairs of multibeam sonar images and poses. To align with the 120° FOV of real sonar data, the original 360° synthetic images were cropped. Different yaw orientations of the AUV were simulated by shifting the crop window proportionally to the desired rotation. During training, we applied five fixed yaw rotations in the range $[-30°, 30°]$ with a 15° step. For validation, we introduced higher variability by applying a continuous random yaw rotation in the same range. The dataset was split 80/20, with 5185 pairs for training and 1296 for validation.

For real-world evaluation, we obtained two datasets of 710 and 790 pairs of sonar images and poses, respectively. Data were acquired in an underwater site containing the same asset modeled in the synthetic environment, simulated with a coarse 3D mesh and the same multibeam sonar type. The real AUV executed two semi-circular trajectories around the asset, shifted by approximately 0.3 m in altitude. Ground truth correspondences between query and database poses were determined by computing the o-IoU between the query pose and all database poses, selecting the one with the highest score. For sonar fan generation, we used a fixed radius $r = 5$ m.

8.4.2 Results

We validated our approach through two experiments.

In the first experiment, we trained the network exclusively on synthetic sonar image–pose pairs and tested it on real sonar data. This setup evaluates the effectiveness of our method and its robustness to domain shifts. Results are reported in the first row of Table 8.1.

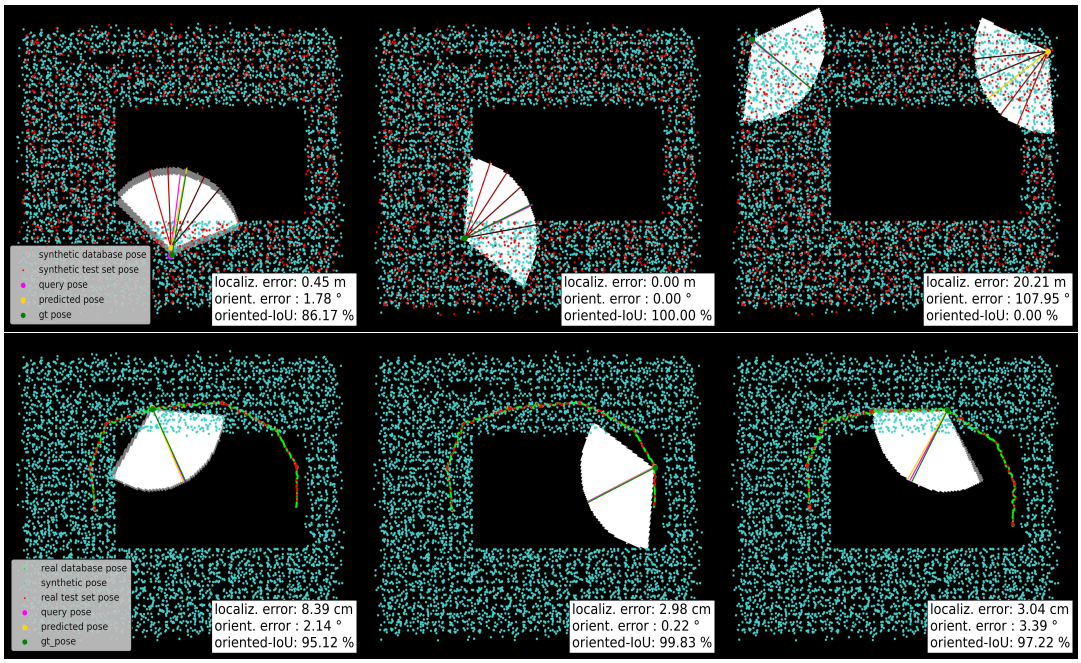


FIGURE 8.3: Six example runs of our system, trained only on synthetic data. A dot represents a position, a line represents the orientation of the AUV. The sonar fan is represented by the circular sector. When two sonar fans are overlapping, white areas represent intersection and grey areas represent exclusion. Otherwise, the area is always white. On the top row, three runs on the synthetic dataset. On the second row, from left to right, three runs on the real datasets. The first two are on the first trajectory, first and fourth folds. The last one is on the second trajectory, first fold.

The second experiment investigates the benefit of incorporating real images. Here, the network was trained to jointly solve two tasks: (i) place recognition and (ii) sonar image reconstruction. Synthetic data were used for both tasks, while real sonar images were used only for reconstruction. The performance of this hybrid model is summarized in the second row of Table 8.1. In both experiments, performance was measured in terms of oriented-IoU (o-IoU) between predicted and ground-truth poses.

For validation, the training set served as the reference database. We computed descriptors for all training images, then queried them with descriptors from the validation set. For testing, we evaluated on two real datasets using a 5-fold cross-validation. Each dataset was randomly split into five folds: one fold served as queries, while the remaining four formed the database. Results were averaged across folds and then across the two datasets to obtain the final performance.

Both experiments used a batch size of 8 and the AdamW optimizer [116] with initial learning rate 0.01, reduced by a factor of 0.8 per epoch. Early stopping was based on validation loss with patience of 10 epochs. Training stopped after 7 epochs in the first experiment and after 43 epochs in the second.

Our results confirm that the proposed network can localize the AUV in both synthetic maps and real underwater environments by comparing sonar descriptors. The larger magnitude of errors in the synthetic dataset is due to the broader range of possible localization outcomes compared to the real datasets.

We also evaluated a randomly weighted network, following [2], which showed that neural networks at initialization can already provide meaningful features. Surprisingly, this baseline achieved performance close to, though consistently worse than, our trained model in the first experiment, and in some cases outperformed in the second experiment on real test sets. However, it consistently lagged behind in the synthetic dataset.

The network requires 10 ms to compute a descriptor and 89 ms to query a database of 25925 descriptors (5185×5, each of length 1024). The total runtime is 99 ms per frame, which satisfies real-time operation at the sonar frame rate of 10 Hz.

8.5 Conclusion

In this chapter, we introduced a novel approach for underwater place recognition using sonar images and robust descriptors. Our method leverages a sonar image descriptor computation, enhanced by an image reconstruction branch to improve feature learning. We also proposed a novel training strategy designed to develop more robust and consistent descriptors. We implemented and evaluated our approach in preliminary experiments, combining synthetic and real data to address domain shifts. The results demonstrate that the proposed network and training strategy are very promising, enabling reliable place recognition in underwater scenarios.

Chapter 9

Conclusions

OBOTIC perception is the key task for autonomous mobile robots to achieve scene understanding, and therefore, enable all the downstream tasks.

With a focus on learning-based perception methods, the main contribution of this thesis is the revision and redesign of traditional methods for traversability analysis, instance and semantic segmentation, place recognition and temporal fruit matching.

In Chapter 3, we develop P-SVM, a geometric-based approach for support vector machine-based traversability analysis comprising a feature fusion module on a pyramid-polar space representation. P-SVM proves to be competitive against state-of-the-art deep learning approaches: it obtains comparable performance but with a lower runtime and power consumption. It runs completely on CPU without the need for high-end GPUs like the other methods. It consumes over 70% less power compared to the deep learning approaches. Additionally, P-SVM is well-suited for embedded devices such as Jetson Xavier board with limited CPU capabilities, enabling quasi-real-time execution. In contrast, other methods fail to meet real-time constraints even when exploiting the embedded GPU.

In Chapter 4 we address the challenge of real-time semantic segmentation of point clouds under both small and large data regimes. We propose WaffleAndRange, combining local feature extraction with hybrid projections of the 3D point cloud and range image, improving both accuracy and efficiency. Experiments on SemanticKITTI and nuScenes confirm that WaffleAndRange generalizes well: it performs strongly with limited data while also remaining competitive in data-rich scenarios. Through GPU-based optimizations and streamlined operations, we reduce runtime significantly, developing a practical solution for autonomous robotics applications.

In Chapter 5 we demonstrate that zero-shot segmentators such as SAM can enhance semantic segmentation when combined with specialized models like DeepLabv3+ and PVTv2. SAM alone does not consistently outperform these models, but checkpoint prompting and, especially, logit-level fusion provide clear improvements across multiple datasets, even surpassing state-of-the-art methods on the datasets CAMO and Butterfly. Limitations include performance variance across domains and medical image segmentation remaining particularly challenging. Ground-truth inaccuracies further complicate evaluation but also highlight the robustness of SAM in some cases.

In Chapter 6 we presented EastPark, a new multi-sensor, large-scale dataset for semantic segmentation and terrain analysis that we collected in a crowded park. It features a unique sensor placement and dense manual annotations of 3D LiDAR point clouds over 22 classes. To establish baselines, we evaluated three state-of-the-art models (PTv3, 3PNet, and LSK3DNet) using official pretrained weights and adapting them with parameter-efficient strategies.

In Chapter 7 we designed a method for temporal fruit monitoring based on fruit instance segmentation and re-identification on colored point clouds acquired by a high-resolution

LiDAR scanner at different points in time. We implement and evaluate our approach on real-world 3D datasets and provide comparisons to other existing techniques. Our method achieves superior performance in both instance segmentation and re-identification, even when trained on ground truth annotations but tested on predicted instance segmentation. This flexibility makes it highly adaptable to different scenarios.

Finally, in Chapter 8 we address robotics perception in extreme conditions, i.e. underwater place recognition using sonar images and robust descriptors. We propose a method that leverages a sonar image descriptor computation, enhanced by an image reconstruction branch to improve feature learning. We also present a novel training strategy designed to develop more robust and consistent descriptors. We evaluate our approach in preliminary experiments, combining synthetic and real data to address domain shifts. Results show the effectiveness of the proposed network and training strategy.

Overall, this thesis presented novel methods to improve the current field of robotic perception, in terms of architectures, training strategies, and datasets. In multiple cases, we exceeded the state-of-the-art in accuracy and runtime, enabling precise and real-time scene understanding. We evaluated most of our proposed methods using publicly available datasets and released most of their implementation to support future research on robotic perception.

Bibliography

[1] Eren Erdal Aksoy, Saimir Baci, and Selcuk Cavdar. SalsaNet: Fast road and vehicle segmentation inLiDAR point clouds for autonomous driving. In *Proc. of the IEEE Vehicles Symposium (IV)*, pages 926–932, 2020.

[2] Ehsan Amid, Rohan Anil, Wojciech Kotłowski, and Manfred K. Warmuth. Learning from randomly initialized neural network features. *arXiv preprint*, 2022.

[3] Angelika Ando, Spyros Gidaris, Andrei Bursuc, Gilles Puy, Alexandre Boulch, and Renaud Marlet. RangeViT: Towards vision transformers for 3D semantic segmentation in autonomous driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5240–5250, 2023.

[4] Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 3D Semantic Parsing of Large-Scale Indoor Spaces. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1534–1543, 2016.

[5] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. SegNet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(12):2481–2495, 2017.

[6] Jens Behley, Martin Garbade, Andres Milioto, Jan Quenzel, Sven Behnke, C. Stachniss, and Juergen Gall. SemanticKITTI: A dataset for semantic scene understanding of LiDAR sequences. *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 9296–9306, 2019.

[7] Mauro Bellone, Giulio Reina, Luca Caltagirone, and Mattias Wahde. Learning traversability from point clouds in challenging scenarios. *IEEE Trans. on Intelligent Transportation Systems (TITS)*, 19:296–305, 2018.

[8] M. Berman, A. R. Triki, and M. B. Blaschko. The Lovász-Softmax Loss: A Tractable Surrogate for the Optimization of the Intersection-Over-Union Measure in Neural Networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4413–4421, 2018.

[9] Christopher M. Bishop. *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2006.

[10] Christopher M Bishop and Hugh Bishop. *Deep Learning: Foundations and Concepts*. Springer Nature, 2023.

[11] Pieter M Blok, Eldert J van Henten, Frits K van Evert, and Gert Kootstra. Image-based size estimation of broccoli heads under varying degrees of occlusion. *Biosystems Engineering*, 208:213–233, 2021.

[12] Frans P. Boogaard, Eldert J. van Henten, and Gert Kootstra. The added value of 3D point clouds for digital plant phenotyping - a case study on internode length measurements in cucumber. *Biosystems Engineering*, 234:1–12, 2023.

[13] L. Brigato and L. Iocchi. A close look at deep learning with small data. In *Proc. of the Intl. Conf. on Pattern Recognition (ICPR)*, pages 2490–2497, 2021.

[14] Pedro O. C. S. Ribeiro, Matheus M. dos Santos, Paulo L. J. Drews, and Silvia S. C. Botelho. Forward looking sonar scene matching using deep learning. In *Proc. of the IEEE Intl. Conf. on Machine Learning And Applications (ICMLA)*, 2017.

[15] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 11618–11628, 2020.

[16] Luca Caltagirone, Samuel Scheidegger, Lennart Svensson, and Mattias Wahde. Fast LiDAR-based road detection using fully convolutional neural networks. In *Proc. of the IEEE Vehicles Symposium (IV)*, pages 1019–1024, 2017.

[17] A. Cardellicchio, F. Solimani, G. Dimauro, S. Summerer, and V. Renò. Patch-based probabilistic identification of plant roots using convolutional neural networks. *Pattern Recognition Letters*, 183:125–132, 2024.

[18] Nicholas Carlevaris-Bianco, Arash K Ushani, and Ryan M Eustice. University of michigan north campus long-term vision and LiDAR dataset. *Intl. Journal of Robotics Research (IJRR)*, 35(9):1023–1035, 2016.

[19] N. Chebrolu, F. Magistri, T. Läbe, and C. Stachniss. Registration of Spatio-Temporal Point Clouds of Plants for Phenotyping. *PLOS ONE*, 16(2), 2021.

[20] Liang Chen, Jian Yang, and Hui Kong. LiDAR-histogram for fast road and obstacle detection. *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1343–1348, 2017.

[21] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 833–851, 2018.

[22] Meida Chen, Qingyong Hu, Zifan Yu, Hugues THOMAS, Andrew Feng, Yu Hou, Kyle McCullough, Fengbo Ren, and Lucio Soibelman. Stpls3d: A large-scale synthetic and real aerial photogrammetry 3D point cloud dataset. In *Proc. of British Machine Vision Conf. (BMVC)*, 2022.

[23] Hui-Xian Cheng, Xian-Feng Han, and Guo-Qiang Xiao. CENet: Toward concise and efficient LiDAR semantic segmentation for autonomous driving. In *Proc. of the IEEE Intl. Conf. on Multimedia and Expo*, pages 01–06, 2022.

[24] Ran Cheng, Ryan Razani, Ehsan Taghavi, Enxu Li, and Bingbing Liu. (AF)2-S3Net: Attentive feature fusion with adaptive feature selection for sparse semantic segmentation network. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 12547–12556, 2021.

[25] François Chollet. Xception: Deep learning with depthwise separable convolutions. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1251–1258, 2017.

[26] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4D spatio-temporal convnets: Minkowski convolutional neural networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3075–3084, 2019.

[27] Pengyu Chu, Zhaojian Li, Kyle Lammers, Renfu Lu, and Xiaoming Liu. Deep learning-based apple detection using a suppression mask R-CNN. *Pattern Recognition Letters*, 147:206–211, 2021.

[28] D. Comaniciu and P. Meer. Mean Shift: A Robust Approach Toward Feature Space Analysis. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 24(5):603–619, 2002.

[29] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 3213–3223, 2016.

[30] Tiago Cortinhal, George Tzelepis, and Eren Erdal Aksoy. SalsaNext: Fast, uncertainty-aware semantic segmentation of LiDAR point clouds. In *Proc. of the Intl. Symposium on Visual Computing (ISVC)*, pages 207–222, 2020.

[31] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, 2000.

[32] Y. Le Cun, B. Boser, J. S. Denker, R. E. Howard, W. Habbard, L. D. Jackel, and D. Henderson. Handwritten digit recognition with a back-propagation network. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, pages 396–404, 1990.

[33] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5828–5839, 2017.

[34] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2009.

[35] Liuyuan Deng, Ming Yang, Zhidong Liang, Yuesheng He, and Chunxiang Wang. Fusing geometrical and visual information via superpoints for the semantic segmentation of 3D road scenes. *Tsinghua Science and Technology*, 25(4):498–507, 2020.

[36] Ivano Donadi, Emilio Olivastri, Daniel Fusaro, Wanmeng Li, Daniele Evangelista, and Alberto Pretto. Improving generalization of synthetically trained sonar image descriptors for underwater place recognition. In *Intl. Conf. on Computer Vision Systems*, 2023.

[37] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2021.

[38] T. Duckett, S. Pearson, S. Blackmore, B. Grieve, W. Chen, G. Cielniak, J. Cleaversmith, J. Dai, S. Davis, C. Fox, P. From, I. Georgilas, R. Gill, I. Gould, M. Hanheide,

A. Hunter, F. Iida, L. Mihalyova, S. Nefti-Meziani, G. Neumann, P. Paoletti, T. Pridmore, D. Ross, M. Smith, M. Stoelen, M. Swainson, S. Wane, P. Wilson, I. Wright, and G. Yang. Agricultural Robotics: The Future of Robotic Agriculture. *arXiv preprint*, arXiv:1806.06762, 2018.

[39] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The PASCAL visual object classes (VOC) challenge. *Intl. Journal of Computer Vision (IJCV)*, 88:303–338, 2010.

[40] Fenglei Fan, Jinjun Xiong, Mengzhou Li, and Ge Wang. On interpretability of artificial neural networks: A survey. *IEEE Trans. on Radiation and Plasma Medical Sciences (TRPMS)*, 5:741–760, 2021.

[41] Ben Fei, Liwen Liu, Weidong Yang, Zhijun Li, Wen-Ming Chen, and Lipeng Ma. Parameter efficient point cloud prompt tuning for unified point cloud understanding. *IEEE Trans. on Intelligent Vehicles (TIV)*, 10(1), 2025.

[42] Jiajun Fei and Zhidong Deng. Fine-tuning point cloud transformers with dynamic aggregation. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 9455–9462, 2024.

[43] Tuo Feng, Wenguan Wang, Fan Ma, and Yi Yang. LSK3DNet: Towards effective and efficient 3D perception with large sparse kernels. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 14916–14927, 2024.

[44] F. Fiorani and U. Schurr. Futurescenarios for plant phenotyping. *Annual Review of Plant Biology*, 64:267–291, 2013.

[45] Daniel Fusaro, Emilio Olivastri, Ivano Donadi, Daniele Evangelista, Emanuele Menegatti, and Alberto Pretto. Pyramidal 3D feature fusion on polar grids for fast and robust traversability analysis on CPU. *Journal on Robotics and Autonomous Systems (RAS)*, 170:104524, 2023.

[46] Daniel Fusaro, Emilio Olivastri, Daniele Evangelista, Pietro Iob, and Alberto Pretto. An hybrid approach to improve the performance of encoder-decoder architectures for traversability analysis in urban environments. In *Proc. of the IEEE Vehicles Symposium (IV)*, pages 1745–1750, 2022.

[47] P. Ganesh, K. Volle, T.F. Burks, and S.S. Mehta. Deep orange: Mask R-CNN based orange detection and segmentation. *IFAC-PapersOnLine*, 52(30):70–75, 2019.

[48] Yuanyue Ge, Ya Xiong, and P@l J. From. Instance segmentation and localization of strawberries in farm conditions for automatic fruit harvesting. *IFAC-PapersOnLine*, 52(30):294–299, 2019.

[49] Jordi Gené-Mola, Ricardo Sanz-Cortiella, Joan R. Rosell-Polo, Alexandre Escolà, and Eduard Gregorio. PFuji-Size dataset: A collection of images and photogrammetry derived 3D point clouds with ground truth annotations for fuji apple detection and size estimation in field conditions. *Data in Brief*, 39:107629, 2021.

[50] Jordi Gené-Mola, Ricardo Sanz-Cortiella, Joan R. Rosell-Polo, Josep-Ramon Morros, Javier Ruiz-Hidalgo, Verónica Vilaplana, and Eduard Gregorio. Fruit detection and 3D location using instance segmentation neural networks and structure from motion photogrammetry. *Computers and Electronics in Agriculture*, 169:105165, 2020.

[51] Edwin Goh, Jingdao Chen, and Brian Wilson. Mars terrain segmentation with less labels. In *Proc. of the IEEE Aerospace Conference (AERO)*, pages 1–10, 2022.

[52] Adrian Salazar Gomez, Erchan Aptoula, Simon Parsons, and Petra Bosilj. Deep regression versus detection for counting in robotic phenotyping. *IEEE Robotics and Automation Letters (RA-L)*, 6(2):2902–2907, 2021.

[53] Sebastian Gonzalez, Claudia Arellano, and Juan E. Tapia. Deepblueberry: Quantification of blueberries in the wild using instance segmentation. *IEEE Access*, 7, 2019.

[54] Benjamin Graham, Martin Engelcke, and Laurens Van Der Maaten. 3D semantic segmentation with submanifold sparse convolutional networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9224–9232, 2018.

[55] G. Grisetti, R. Kümmerle, C. Stachniss, and W. Burgard. A tutorial on graph-based SLAM. *IEEE Trans. on Intelligent Transportation Systems Magazine*, 2(4):31–43, 2010.

[56] Tiziano Guadagnino, Benedikt Mersch, Ignacio Vizzo, Saurabh Gupta, Meher VR Malladi, Luca Lobefaro, Guillaume Doisy, and Cyrill Stachniss. Kinematic-ICP: Enhancing LiDAR odometry with kinematic constraints for wheeled mobile robots moving on planar surfaces. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 11090–11096, 2025.

[57] Tianrui Guan, Zhenpeng He, Dinesh Manocha, and Liangjun Zhang. TTM: Terrain traversability mapping for autonomous excavator navigation in unstructured environments. *arXiv preprint*, 2021.

[58] Yulan Guo, Hanyun Wang, Qingyong Hu, Hao Liu, Li Liu, and Mohammed Bennamoun. Deep learning for 3D point clouds: A survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 43(12):4338–4364, 2020.

[59] T Hackel, N Savinov, L Ladicky, JD Wegner, K Schindler, and M Pollefeys. Semantic3D. net: A new large-scale point cloud classification benchmark. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4:91–98, 2017.

[60] Michael Halstead, Christopher McCool, Simon Denman, Tristan Perez, and Clinton Fookes. Fruit quantity and ripeness estimation using a robotic vision system. *IEEE Robotics and Automation Letters (RA-L)*, 3(4):2995–3002, 2018.

[61] Haoyuan Hao, Sheng Wu, YuanKun Li, Weiliang Wen, jiangchuan Fan, Yongjiang Zhang, Lvhan Zhuang, Longqin Xu, Hongxin Li, Xinyu Guo, and Shuangyin Liu. Automatic acquisition, analysis and wilting measurement of cotton 3D phenotype based on point cloud. *Biosystems Engineering*, 239:173–189, 2024.

[62] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask R-CNN. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 2017.

[63] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[64] Noriaki Hirose, Amir Sadeghian, Marynel Vázquez, Patrick Goebel, and Silvio Savarese. GONet: A semi-supervised deep learning approach for traversability estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 3044–3051, 2018.

[65] Noriaki Hirose, Amir Sadeghian, Fei Xia, Roberto Martin-Martin, and Silvio Savarese. VUNet: Dynamic scene view synthesis for traversability estimation using an RGB camera. *IEEE Robotics and Automation Letters (RA-L)*, 4(2):2062–2069, 2019.

[66] Sadegh Hosseinpoor, Jim Torresen, Mathias Mantelli, Diego Pitto, Mariana Kolberg, Renan Maffei, and Edson Prestes. Traversability analysis by semantic terrain segmentation for mobile robots. In *Proc. of the Intl. Conf. on Automation Science and Engineering (CASE)*, pages 1407–1413, 2021.

[67] Yuenan Hou, Xinge Zhu, Yuexin Ma, Chen Change Loy, and Yikang Li. Point-to-voxel knowledge distillation for LiDAR semantic segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 8479–8488, 2022.

[68] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. LoRA: Low-rank adaptation of large language models. *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 1(2):3, 2022.

[69] Qingyong Hu, Bo Yang, Linhai Xie, Stefano Rosa, Yulan Guo, Zhihua Wang, Niki Trigoni, and Andrew Markham. RandLA-Net: Efficient semantic segmentation of large-scale point clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 11108–11117, 2020.

[70] Forrest N Iandola, Song Han, Matthew W Moskewicz, Khalid Ashraf, William J Dally, and Kurt Keutzer. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and <0.5 MB model size. *arXiv preprint*, 2016.

[71] S. Ioffe and C. Szegedy. Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift. In *Proc. of the Intl. Conf. on Machine Learning (ICML)*, 2015.

[72] Weikuan Jia, Zhonghua Zhang, Wenjing Shao, Sujuan Hou, Ze Ji, Guoliang Liu, and Xiang Yin. FoveaMask: A fast and accurate deep learning model for green fruit instance segmentation. *Computers and Electronics in Agriculture*, 191:106488, 2021.

[73] Peng Jiang, Philip Osteen, Maggie Wigness, and Srikanth Saripalli. RELLIS-3D dataset: Data, benchmarks and analysis. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1110–1116, 2021.

[74] Ian T. Jolliffe and Jorge Cadima. Principal component analysis: A review and recent developments. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 374, 2016.

[75] Hervé Jégou, Matthijs Douze, Cordelia Schmid, and Patrick Pérez. Aggregating local descriptors into a compact image representation. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2010.

[76] Hanwen Kang and Chao Chen. Fruit detection, segmentation and 3D visualisation of environments in apple orchards. *Computers and Electronics in Agriculture*, 171:105302, 2020.

[77] Hanwen Kang, Xing Wang, and Chao Chen. Accurate fruit localisation using high resolution LiDAR-camera fusion and instance segmentation. *Computers and Electronics in Agriculture*, 203:107450, 2022.

[78] Lei Ke, Mingqiao Ye, Martin Danelljan, Yu-Wing Tai, Chi-Keung Tang, Fisher Yu, et al. Segment anything in high quality. *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 36:29914–29934, 2023.

[79] Alex Kendall, Matthew Grimes, and Roberto Cipolla. PoseNet: A convolutional network for real-time 6-DOF camera relocalization. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 2938–2946, 2015.

[80] Jana Kierdorf, Immanuel Weber, Anna Kicherer, Laura Zabawa, Lukas Drees, and Ribana Roscher. Behind the leaves: Estimation of occluded grapevine berries with conditional generative adversarial networks. *Frontiers in Artificial Intelligence*, 5, 2022.

[81] Donghoon Kim, Donghwa Lee, Hyun Myung, and Hyun-Taek Choi. Artificial landmark based underwater localization for AUVs using weighted template matching. *Intelligent Service Robotics*, 7:175–184, 2014.

[82] Kyuwon Kim, Beomjun Kim, Kyoung-Jun Lee, Bongchul Ko, and Kyongsu Yi. Design of integrated risk management-based dynamic driving control of automated vehicles. *IEEE Trans. on Intelligent Transportation Systems Magazine*, 9:57–73, 2017.

[83] Yong-Woon Kim, Yung-Cheol Byun, and Addapalli V. N. Krishna. Portrait segmentation using ensemble of heterogeneous deep-learning models. *Entropy*, 23(2), 2021.

[84] A. Kirillov, R. Girshick, K. He, and P. Dollar. Panoptic Feature Pyramid Networks. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 6399–6408, 2019.

[85] A. Kirillov, K. He, R. Girshick, C. Rother, and P. Dollár. Panoptic Segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9404–9413, 2019.

[86] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 4015–4026, 2023.

[87] Josef Kittler. Combining classifiers: A theoretical framework. *Pattern Analysis and Applications*, 1:18–27, 1998.

[88] Deyvid Kochanov, Fatemeh Karimi Nejadasl, and Olaf Booij. Kprnet: Improving projection-based LiDAR semantic segmentation. *arXiv preprint*, 2020.

[89] Maxim Kolodiazhnyi, Anna Vorontsova, Anton Konushin, and Danila Rukhovich. OneFormer3D: One transformer for unified point cloud segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 20943–20953, 2024.

[90] Lingdong Kong, Youquan Liu, Runnan Chen, Yuexin Ma, Xinge Zhu, Yikang Li, Yuenan Hou, Yu Qiao, and Ziwei Liu. Rethinking range view representation for LiDAR segmentation. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 228–240, 2023.

[91] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, volume 25, 2012.

[92] Ludmila I. Kuncheva. Diversity in multiple classifier systems. *Information Fusion*, 6(1):3–4, 2005.

[93] Xin Lai, Yukang Chen, Fanbin Lu, Jianhui Liu, and Jiaya Jia. Spherical transformer for LiDAR based 3D recognition. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 17545–17555, 2023.

[94] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4558–4567, 2018.

[95] Loic Landrieu and Martin Simonovsky. Large-scale point cloud semantic segmentation with superpoint graphs. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4558–4567, 2018.

[96] Max Larsson, Nils Bore, and John Folkesson. Latent space metric learning for sidescan sonar place recognition. In *IEEE/OES Autonomous Underwater Vehicles (AUV) Symposium*, 2020.

[97] Trung-Nghia Le, Tam V. Nguyen, Zhongliang Nie, Minh-Triet Tran, and Akihiro Sugimoto. Anabranch network for camouflaged object segmentation. *Journal of Computer Vision and Image Understanding (CVIU)*, 184:45–56, 2019.

[98] Daehan Lee, Hyungtae Lim, and Soohee Han. GenZ-ICP: Generalizable and degeneracy-robust LiDAR odometry using an adaptive weighting. *IEEE Robotics and Automation Letters (RA-L)*, 2024.

[99] Hyunsuk Lee and Woojin Chung. A self-training approach-based traversability analysis for mobile robots in urban environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3389–3394, 2021.

[100] John J. Leonard, Jonathan P. How, Seth J. Teller, Mitch Berger, Stefan Campbell, Gaston A. Fiore, Luke Fletcher, Emilio Frazzoli, Albert S. Huang, Sertaç Karaman, Olivier Koch, Yoshiaki Kuwata, David C. Moore, Edwin Olson, Steven C. Peters, Justin Teo, Robert Truax, Matthew R. Walter, David Barrett, Alexander Epstein, Keoni Maheloni, Katy Moyer, Troy Jones, Ryan Buckley, Matthew E. Antone, Robert Galejs, Siddhartha Krishnamurthy, and Jonathan Williams. A perception-driven autonomous urban vehicle. In *The DARPA Urban Challenge*, pages 163–230, 2009.

[101] Haibin Li, Guangwei Wang, Xiaohui Li, and Yue Lian. A review of underwater SLAM technologies. In *IEEE Intl. Conf. on Robotics, Intelligent Control and Artificial Intelligence (RICAI)*, pages 215–225, 2023.

[102] Jie Li, Michael Kaess, Ryan M Eustice, and Matthew Johnson-Roberson. Pose-graph SLAM using forward looking sonar. *IEEE Robotics and Automation Letters (RA-L)*, 3(3):2330–2337, 2018.

[103] Jie Li, Paul Ozog, Jacob Abernethy, Ryan M. Eustice, and Matthew Johnson-Roberson. Utilizing high dimensional features for real time robotic applications reducing the curse of dimensionality for recursive bayesian estimation. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2016.

[104] Mengke Li, Da Li, Guoqing Yang, Yiu-ming Cheung, and Hui Huang. Adapt PointFormer: 3D point cloud analysis via adapting 2D visual transformers. In *Proc. of the Europ. Conf. on European Conference on Artificial Intelligence (ECAI)*, pages 89–96, 2024.

[105] Chen Liang, Wenguan Wang, Jiaxu Miao, and Yi Yang. GMMSeg: Gaussian mixture based generative semantic segmentation models. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, volume 35, pages 31360–31375, 2022.

[106] Zhidong Liang, Ming Yang, Liuyuan Deng, Chunxiang Wang, and Bing Wang. Hierarchical depthwise graph convolutional neural network for 3D semantic segmentation of point clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 8152–8158, 2019.

[107] Zhidong Liang, Ming Yang, Hao Li, and Chunxiang Wang. 3D instance embedding learning with a structure-aware loss function for point cloud segmentation. *IEEE Robotics and Automation Letters (RA-L)*, 5(3):4915–4922, 2020.

[108] Yiyi Liao, Jun Xie, and Andreas Geiger. KITTI-360: A novel dataset and benchmarks for urban scene understanding in 2D and 3D. *arXiv preprint*, 2021.

[109] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 740–755, 2014.

[110] Lei Liu, Guorun Li, Yuefeng Du, Xiaoyu Li, Xiuheng Wu, Zhi Qiao, and Tianyi Wang. CS-Net: Conv-SimpleFormer network for agricultural image segmentation. *Pattern Recognition*, 147:110140, 2024.

[111] Limei Liu, Meng Liu, Kexin Meng, Linqiang Yang, Min Zhao, and Shuli Mei. Camouflaged locust segmentation based on PraNet. *Computers and Electronics in Agriculture*, 198:107061, 2022.

[112] Youquan Liu, Runnan Chen, Xin Li, Lingdong Kong, Yuchen Yang, Zhaoyang Xia, Yeqi Bai, Xinge Zhu, Yuexin Ma, Yikang Li, et al. UniSeg: A unified multi-modal LiDAR segmentation network and the openpcseg codebase. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 21662–21673, 2023.

[113] Yuanzhi Liu, Yujia Fu, Minghui Qin, Yufeng Xu, Baoxin Xu, Fengdong Chen, Bart Goossens, Poly ZH Sun, Hongwei Yu, Chun Liu, et al. BotanicGarden: A high quality dataset for robot navigation in unstructured natural environments. *IEEE Robotics and Automation Letters (RA-L)*, 9(3):2798–2805, 2024.

[114] Luca Lobefaro, Meher VR Malladi, Tiziano Guadagnino, and Cyrill Stachniss. Spatio-temporal consistent mapping of growing plants for agricultural robots in the wild. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 6375–6382, 2024.

[115] Luca Lobefaro, Meher VR Malladi, Olga Vysotska, Tiziano Guadagnino, and Cyrill Stachniss. Estimating 4D data associations towards spatial temporal mapping of growing plants for agricultural robots. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4212–4218, 2023.

[116] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *Proc. of the Intl. Conf. on Learning Representations (ICLR)*, 2019.

[117] D.G. Lowe. Object recognition from local scale-invariant features. In *Proc. of the IEEE Intl. Conf. on Computer Vision (ICCV)*, 1999.

[118] Alessandra Lumini and Loris Nanni. Fair comparison of skin detection approaches on publicly available datasets. *Expert Systems with Applications*, 160:113677, 2020.

[119] Hanjiang Luo, Yiyang Zhao, Zhongwen Guo, Siyuan Liu, Pengpeng Chen, and Lionel M. Ni. UDB: Using directional beacons for localization in underwater sensor networks. In *IEEE Intl. Conf. on Parallel and Distributed Systems*, 2008.

[120] F. Magistri, Y. Pan, J. Bartels, J. Behley, C. Stachniss, and C. Lehnert. Improving Robotic Fruit Harvesting Within Cluttered Environments Through 3D Shape Completion. *IEEE Robotics and Automation Letters (RA-L)*, 9(8):7357–7364, 2024.

[121] Alejandro Maldonado-Ramírez and L. Abril Torres-Mendez. Learning ad-hoc compact representations from salient landmarks for visual place recognition in underwater environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2019.

[122] R. Marcuzzi, L. Nunes, L. Wiesmann, J. Behley, and C. Stachniss. Mask-Based Panoptic LiDAR Segmentation for Autonomous Driving. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1141–1148, 2023.

[123] Daniel Maturana, Po-Wei Chou, Masashi Uenoyama, and Sebastian Scherer. Real-time semantic mapping for autonomous off road navigation. In *Field and Service Robotics*, pages 335–350, 2017.

[124] Francesco Maurelli, Szymon Krupiński, Xianbo Xiang, and Yvan Petillot. AUV localisation a review of passive and active techniques. *Intl. Journal of Intelligent Robotics and Applications*, 6:246–269, 2022.

[125] Francesco Maurelli, Yvan Petillot, Angelos Mallios, Pere Ridao, and Szymon Krupinski. Sonar-based AUV localization using an improved particle filter approach. In *OCEANS - Europe*, pages 1–9, 2009.

[126] John McConnell, Fanfei Chen, and Brendan Englot. Overhead image factors for underwater sonar based SLAM. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):4901–4908, 2022.

[127] Gledson Melotti, Cristiano Premebida, Nuno M. M. da S. Goncalves, Urbano J. C. Nunes, and Diego R. Faria. Multimodal CNN pedestrian classification: A study on combining lidar and camera data. In *Proc. of the IEEE Intl. Conf. on Intelligent Transportation Systems (ITSC)*, pages 3138–3143, 2018.

[128] B. Mildenhall, P.P. Srinivasan, M. Tancik, J.T. Barron, R. Ramamoorthi, and R. Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 405–421, 2020.

[129] Andres Milioto, Ignacio Vizzo, Jens Behley, and C. Stachniss. RangeNet++: Fast and accurate LiDAR semantic segmentation. *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4213–4220, 2019.

[130] Peter Mortimer, Raphael Hagmanns, Miguel Granero, Thorsten Luettel, Janko Petereit, and Hans-Joachim Wuensche. The GOOSE dataset for perception in unstructured environments. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 14838–14844, 2024.

[131] Simone Mosco, Daniel Fusaro, Wanmeng Li, Emanuele Menegatti, and Alberto Pretto. Point-plane projections for accurate LiDAR semantic segmentation in small data scenarios. *arXiv preprint*, 2025.

[132] Loris Nanni, Sheryl Brahnam, and Alessandra Lumini. Ensemble of deep learning approaches for atc classification. In *Proc. of the Intl. Conf. on Smart Computing and Informatics (SCI)*, pages 117–125, 2020.

[133] Loris Nanni, Alessandra Lumini, Andrea Loreggia, Alberto Formaggio, and Daniela Cuza. An empirical study on ensemble of segmentation approaches. *Signals*, 3(2):341–358, 2022.

[134] Hoang C. Nguyen, Tung T. Le, Hieu H. Pham, and Ha Q. Nguyen. VinDr-RibCXR: A benchmark dataset for automatic segmentation and labeling of individual ribs on chest X-rays. In *Proc. of the Intl. Conf. on Medical Imaging with Deep Learning (MIDL)*, 2021.

[135] S. Nuske, S. Achar, T. Bates, S. Narasimhan, and S. Singh. Yield Estimation in Vineyards by Visual Grape Detection. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, 2011.

[136] A. Oliva and A. Torralba. Building the gist of a scene: The role of global image features in recognition. *Progress in Brain Research*, 155:23–36, 2006.

[137] Emilio Olivastri, Daniel Fusaro, Wanmeng Li, Simone Mosco, and Alberto Pretto. A sonar-based AUV positioning system for underwater environments with low infrastructure density. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2024.

[138] Edwin Olson, John J. Leonard, and Seth Teller. Robust range-only beacon localization. *IEEE Journal of Oceanic Engineering*, 31(4):949–958, 2006.

[139] Yancheng Pan, Biao Gao, Jilin Mei, Sibo Geng, Chengkun Li, and Huijing Zhao. SemanticPOSS: A point cloud dataset with large quantity of dynamic instances. In *Proc. of the IEEE Vehicles Symposium (IV)*, pages 687–693, 2020.

[140] Panagiotis Papadakis. Terrain traversability analysis methods for unmanned ground vehicles: A survey. *Engineering Applications of Artificial Intelligence*, 26(4):1373–1385, 2013.

[141] Jaehyun Park, Chansoo Kim, Soyeong Kim, and Kichun Jo. PCSCNet: Fast 3D semantic segmentation of LiDAR point cloud for autonomous car using point convolution and sparse convolution network. *Expert Systems with Applications*, 212:118815, 2023.

[142] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An imperative style, high-performance deep learning library. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, pages 8024–8035, 2019.

[143] David Paz, Hengyuan Zhang, Qinru Li, Hao Xiang, and Henrik I. Christensen. Probabilistic semantic mapping for urban autonomous driving applications. *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 2059–2064, 2020.

[144] Giuseppe Peano. Sur une courbe, qui remplit toute une aire plane. In *Arbeiten zur Analysis und zur mathematischen Logik*, pages 71–75, 1890.

[145] Gilles Puy, Alexandre Boulch, and Renaud Marlet. Using a waffle iron for automotive point cloud semantic segmentation. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 3379–3389, 2023.

[146] Isaac Pérez-Borrero, Diego Marín-Santos, Manuel E. Gegúndez-Arias, and Estefanía Cortés-Ancos. A fast and accurate deep learning method for strawberry instance segmentation. *Computers and Electronics in Agriculture*, 178:105736, 2020.

[147] C. Qi, Hao Su, Kaichun Mo, and Leonidas J. Guibas. PointNet: Deep learning on point sets for 3D classification and segmentation. *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 77–85, 2017.

[148] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 30, 2017.

[149] Sharmin Rahman, Alberto Quattrini Li, and Ioannis Rekleitis. SVIn2: A multi sensor fusion based underwater slam system. *Intl. Journal of Robotics Research (IJRR)*, 41(11-12):1022–1042, 2022.

[150] Ryan Razani, Ran Cheng, Ehsan Taghavi, and Liu Bingbing. Lite-HDSeg: LiDAR semantic segmentation using lite harmonic dense convolutions. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 9550–9556, 2021.

[151] Pedro O. C. S. Ribeiro, Matheus M. dos Santos, Paulo L. J. Drews, Silvia S. C. Botelho, Lucas M. Longaray, Giovanni G. Giacomo, and Marcelo R. Pias. Underwater place recognition in unknown environments with triplet based acoustic image retrieval. In *Proc. of the IEEE Intl. Conf. on Machine Learning And Applications (ICMLA)*, 2018.

[152] A. Riccardi, S. Kelly, E. Marks, F. Magistri, T. Guadagnino, J. Behley, M. Bennewitz, and C. Stachniss. Fruit Tracking Over Time Using High-Precision Point Clouds. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 9630–9636, 2023.

[153] Damien Robert, Hugo Raguet, and Loic Landrieu. Efficient 3D semantic segmentation with superpoint transformer. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 17195–17204, 2023.

[154] Damien Robert, Hugo Raguet, and Loic Landrieu. Scalable 3D panoptic segmentation as superpoint graph clustering. *Proc. of the Intl. Conf. on 3D Vision (3DV)*, pages 179–189, 2024.

[155] Javier Rodriguez-Sanchez, Kyle Johnsen, and Changying Li. A ground mobile robot for autonomous terrestrial laser scanning-based field phenotyping. *arXiv preprint*, 2024.

[156] Jhonn Pablo Rodríguez, David Camilo Corrales, Jean-Noël Aubertot, and Juan Carlos Corrales. A computer vision system for automatic cherry beans detection on coffee trees. *Pattern Recognition Letters*, 136:142–153, 2020.

[157] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In *Proc. of the Intl. Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 234–241, 2015.

[158] Georgios Salavasidis, Andrea Munafò, Catherine A. Harris, Thomas Prampart, Robert Templeton, Micheal Smart, Daniel T. Roper, Miles Pebody, Stephen D. McPhail, Eric Rogers, and Alexander B. Phillips. Terrain aided navigation for long endurance and deep rated autonomous underwater vehicles. *Journal of Field Robotics (JFR)*, 36(2):447–474, 2019.

[159] Matheus M. Santos, Guilherme B. Zaffari, Pedro O. C. S. Ribeiro, Paulo L. J. Drews-Jr, and Silvia S. C. Botelho. Underwater place recognition using forward-looking sonar images: A topological approach. *Journal of Field Robotics (JFR)*, 36(2):355–369, 2019.

[160] Stefan Schubert, Peer Neubert, Sourav Garg, Michael Milford, and Tobias Fischer. Visual place recognition: A tutorial [tutorial]. *IEEE Robotics and Automation Magazine (RAM)*, 31(3):139–153, 2024.

[161] Jonas Schult, Francis Engelmann, Alexander Hermans, Or Litany, Siyu Tang, and Bastian Leibe. Mask3D: Mask transformer for 3D semantic instance segmentation. *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 8216–8223, 2023.

[162] Tal Shaharabany, Aviad Dahan, Raja Giryes, and Lior Wolf. Autosam: Adapting SAM to medical images by overloading the prompt encoder, 2023.

[163] Tixiao Shan and Brendan Englot. LeGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4758–4765, 2018.

[164] Tixiao Shan, Brendan Englot, Drew Meyers, Wei Wang, Carlo Ratti, and Daniela Rus. LIO-SAM: Tightly-coupled LIDAR inertial odometry via smoothing and mapping. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5135–5142, 2020.

[165] Tixiao Shan, Jinkun Wang, Brendan Englot, and Kevin Doherty. Bayesian generalized kernel inference for terrain traversability mapping. In *Proc. of the Conf. on Robot Learning (CoRL)*, volume 87, pages 829–838, 2018.

[166] Evan Shelhamer, Jonathan Long, and Trevor Darrell. Fully convolutional networks for semantic segmentation. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 39(4):640–651, 2017.

[167] Sangyun Shin, Kaichen Zhou, Madhu Vankadari, Andrew Markham, and Niki Trigoni. Spherical mask: Coarse-to-fine 3D point cloud instance segmentation with spherical representation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 4060–4069, 2024.

[168] Kavan Singh Sikand, Sadegh Rabiee, Adam Uccello, Xuesu Xiao, Garrett Warnell, and Joydeep Biswas. Visual representation learning for preference-aware path planning. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 11303–11309, 2022.

[169] Luan Silveira, Felipe Guth, Paulo Drews-Jr, Pedro Ballester, Matheus Machado, Felipe Codevilla, Nelson Duarte-Filho, and Silvia Botelho. An open-source bio-inspired solution to underwater SLAM. *IFAC-PapersOnLine*, 48(2):212–217, 2015.

[170] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[171] Sayanan Sivaraman and Mohan Manubhai Trivedi. Dynamic probabilistic drivability maps for lane change and merge driver assistance. *IEEE Trans. on Intelligent Transportation Systems (TITS)*, 15:2063–2073, 2014.

[172] Claus Smitt, Michael Halstead, Tobias Zaenker, Maren Bennewitz, and Chris McCool. PATHoBot: A robot for glasshouse crop phenotyping and intervention. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, 2021.

[173] Carole H. Sudre, Wenqi Li, Tom Vercauteren, Sebastien Ourselin, and Jorge M. Cardoso. Generalised Dice overlap as a deep learning loss function for highly unbalanced segmentations. In *Proc. of the Intl. Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 240–248, 2017.

[174] Benjamin Suger, Bastian Steder, and Wolfram Burgard. Traversability analysis for mobile robots in outdoor environments: A semi-supervised learning approach based on 3D-LiDAR data. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 3941–3946, 2015.

[175] Hongyu Sun, Yongcai Wang, Wang Chen, Haoran Deng, and Deying Li. Parameter-efficient prompt learning for 3D point cloud understanding. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 9478–9486, 2024.

[176] Jiahao Sun, Chunmei Qing, Junpeng Tan, and Xiangmin Xu. Superpoint transformer for 3D scene instance segmentation. In *Proc. of the Conf. on Advancements of Artificial Intelligence (AAAI)*, volume 37, pages 2393–2401, 2023.

[177] Pei Sun, Henrik Kretzschmar, Xerxes Dotiwalla, Aurelien Chouard, Vijaysai Patnaik, Paul Tsui, James Guo, Yin Zhou, Yuning Chai, Benjamin Caine, et al. Scalability in perception for autonomous driving: Waymo open dataset. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 2446–2454, 2020.

[178] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 1–9, 2015.

[179] Mingxing Tan and Quoc Le. EfficientNet: Rethinking model scaling for convolutional neural networks. In *International conference on machine learning*, pages 6105–6114, 2019.

[180] Haotian Tang, Zhijian Liu, Shengyu Zhao, Yujun Lin, Ji Lin, Hanrui Wang, and Song Han. Searching efficient 3D architectures with sparse point-voxel convolution. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 685–702, 2020.

[181] Shixi Tang, Zilin Xia, Jinan Gu, Wenbo Wang, Zedong Huang, and Wenhao Zhang. High-precision apple recognition and localization method based on RGB-D and improved SOLOv2 instance segmentation. *Frontiers in Sustainable Food Systems*, 8, 2024.

[182] Yiwen Tang, Ray Zhang, Zoey Guo, Xianzheng Ma, Bin Zhao, Zhigang Wang, Dong Wang, and Xuelong Li. Point-PEFT: Parameter-efficient fine-tuning for 3D pre-trained models. In *Proc. of the Conf. on Advancements of Artificial Intelligence (AAAI)*, pages 5171–5179, 2024.

[183] Yiwen Tang, Ray Zhang, Jiaming Liu, Zoey Guo, Bin Zhao, Zhigang Wang, Peng Gao, Hongsheng Li, Dong Wang, and Xuelong Li. Any2Point: Empowering any-modality large models for efficient 3D understanding. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 456–473, 2024.

[184] Lyne Tchapmi, Christopher Choy, Iro Armeni, JunYoung Gwak, and Silvio Savarese. SEGCloud: Semantic segmentation of 3D point clouds. In *Proc. of the Intl. Conf. on 3D Vision (3DV)*, pages 537–547, 2017.

[185] Ma Teng, Li Ye, Zhao Yuxin, Jiang Yanqing, Cong Zheng, Zhang Qiang, and Xu Shuo. An AUV localization and path planning algorithm for terrain aided navigation. *ISA Transactions*, 103:215–227, 2020.

[186] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. KPConv: Flexible and deformable convolution for point clouds. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 6411–6420, 2019.

[187] S. Thrun, W. Burgard, and D. Fox. *Probabilistic Robotics*. MIT Press, 2005.

[188] Sebastian Thrun, Michael Montemerlo, Hendrik Dahlkamp, David Stavens, Andrei Aron, James Diebel, Philip Fong, John Gale, Morgan Halpenny, Gabriel Hoffmann, Kenny Lau, Celia Oakley, Mark Palatucci, Vaughan Pratt, Pascal Stang, Sven Strohband, Cedric Dupont, Lars-Erik Jendrossek, Christian Koelen, and Pamela Mahoney. Stanley: The robot that won the DARPA grand challenge. *Journal of Field Robotics (JFR)*, 23:661–692, 2006.

[189] Nina Varney, Vijayan K Asari, and Quinn Graehling. DALES: A large-scale aerial LiDAR data set for semantic segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition Workshops*, pages 186–187, 2020.

[190] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 30, 2017.

[191] Kavisha Vidanapathirana, Joshua Knights, Stephen Hausler, Mark Cox, Milad Ramezani, Jason Jooste, Ethan Griffiths, Shaheer Mohamed, Sridha Sridharan, Clinton Fookes, et al. WildScenes: A benchmark for 2D and 3D semantic segmentation in large scale natural environments. *Intl. Journal of Robotics Research (IJRR)*, 44(4):532–549, 2025.

[192] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, Stéfan J. van der Walt, Matthew Brett, Joshua Wilson, K. Jarrod Millman, Nikolay Mayorov, Andrew R. J. Nelson, Eric Jones, Robert Kern, Eric Larson, C J Carey, İlhan Polat, Yu Feng, Eric W. Moore, Jake VanderPlas, Denis Laxalde, Josef Perktold, Robert Cimrman, Ian Henriksen, E. A. Quintero, Charles R. Harris, Anne M. Archibald, Antônio H. Ribeiro, Fabian Pedregosa, Paul van Mulbregt, and SciPy 1.0 Contributors. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, 17:261–272, 2020.

[193] K. S. Viswanath, Kartikeya Singh, Peng Jiang, P. B. Sujit, and Srikanth Saripalli. OFFSEG: A semantic segmentation framework for off-road driving. *Proc. of the Intl. Conf. on Automation Science and Engineering (CASE)*, pages 354–359, 2021.

[194] Ignacio Vizzo, Tiziano Guadagnino, Benedikt Mersch, Louis Wiesmann, Jens Behley, and Cyrill Stachniss. KISS-ICP: In defense of point-to-point ICP – simple, accurate, and robust registration if done the right way. *IEEE Robotics and Automation Letters (RA-L)*, 8(2):1029–1036, 2023.

[195] A. Walter, R. Finger, R. Huber, and N. Buchmann. Opinion: Smart farming is key to developing sustainable agriculture. *Proc. of the National Academy of Sciences*, 114(24):6148–6150, 2017.

[196] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, Wenyu Liu, and Bin Xiao. Deep high-resolution representation learning for visual recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence (TPAMI)*, 43(10):3349–3364, 2021.

[197] Jinkun Wang, Fanfei Chen, Yewei Huang, John McConnell, Tixiao Shan, and Brendan Englot. Virtual maps for autonomous exploration of cluttered underwater environments. *IEEE Journal of Oceanic Engineering*, 47(4):916–935, 2022.

[198] Josiah Wang, Katja Markert, and Mark Everingham. Learning models for object recognition from natural language descriptions. In *British Machine Vision Conference (BMVC)*, 2009.

[199] Song Wang, Xiaolu Liu, Lingdong Kong, Jianyun Xu, Chunyong Hu, Gongfan Fang, Wentong Li, Jianke Zhu, and Xinchao Wang. PointLoRA: Low-rank adaptation with token selection for point cloud learning. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 6605–6615, 2025.

[200] Wenguan Wang, Tianfei Zhou, Fisher Yu, Jifeng Dai, Ender Konukoglu, and Luc Van Gool. Exploring cross-image pixel contrast for semantic segmentation. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 7283–7293, 2021.

[201] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pyramid vision transformer: A versatile backbone for dense prediction without convolutions. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 568–578, 2021.

[202] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. PVT v2: Improved baselines with pyramid vision transformer. *Computational Visual Media*, 8(3):415–424, 2022.

[203] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. SOLOv2: Dynamic and fast instance segmentation. *Advances in Neural Information Processing Systems*, 2020.

[204] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E. Sarma, Michael M. Bronstein, and Justin M. Solomon. Dynamic graph CNN for learning on point clouds. *ACM Trans. on Graphics (TOG)*, 38(5), 2019.

[205] Michelle Watt, Fabio Fiorani, Björn Usadel, Uwe Rascher, Onno Muller, and Ulrich Schurr. Phenotyping: New windows into the plant for breeders. *Annual Review of Plant Biology*, 71(1), 2020.

[206] Maggie Wigness, Sungmin Eum, John G Rogers, David Han, and Heesung Kwon. A RUGD dataset for autonomous navigation and visual perception in unstructured outdoor environments. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 5000–5007, 2019.

[207] Eric Willemenot, Pierre-Yves Morvan, Hubert Pelletier, and Arne Hoof. Subsea positioning by merging inertial and acoustic technologies. In *OCEANS - Europe*, pages 1–8, 2009.

[208] Li Wm, D Fusaro, E Olivastri, S Mosco, N Bellotto, A Pretto, et al. P-SVM2: Enhancing LiDAR-based traversability analysis with augmented point cloud descriptor for autonomous mobile systems. In *Proc. of the IEEE Intl. Conf. on Cybernetics and Intelligent Systems (CIS), Robotics, Automation and Mechatronics (RAM)*, pages 352–359, 2024.

[209] Daniel Wolf, Johann Prankl, and Markus Vincze. Fast semantic segmentation of 3D point clouds using a dense CRF with learned parameters. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 4867–4873, 2015.

[210] Bichen Wu, Alvin Wan, Xiangyu Yue, and Kurt Keutzer. SqueezeSeg: Convolutional neural nets with recurrent CRF for real-time road-object segmentation from 3D LiDAR point cloud. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 1887–1893, 2018.

[211] Bichen Wu, Xuanyu Zhou, Sicheng Zhao, Xiangyu Yue, and Kurt Keutzer. SqueezeSegV2: Improved model structure and unsupervised domain adaptation for road-object segmentation from a LiDAR point cloud. In *Proc. of the IEEE Intl. Conf. on Robotics & Automation (ICRA)*, pages 4376–4382, 2019.

[212] Houde Wu, Yuchen Hou, Wenhai Xu, and Ming Zhao. Ultra low light level digital still camera for autonomous underwater vehicle. *Optical Engineering*, 58(1):013106, 2019.

[213] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3D point clouds. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9621–9630, 2019.

[214] Xiaoyang Wu, Li Jiang, Peng-Shuai Wang, Zhijian Liu, Xihui Liu, Yu Qiao, Wanli Ouyang, Tong He, and Hengshuang Zhao. Point transformer v3: Simpler, faster, stronger. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2024.

[215] Xiaoyang Wu, Yixing Lao, Li Jiang, Xihui Liu, and Hengshuang Zhao. Point transformer v2: Grouped vector attention and partition-based pooling. *Advances in Neural Information Processing Systems*, 35:33330–33342, 2022.

[216] Binbin Xiang, Yuanwen Yue, Torben Peters, and Konrad Schindler. A review of panoptic segmentation for mobile mapping point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing (JPRS)*, 203:373–391, 2023.

[217] Pengchuan Xiao, Zhenlei Shao, Steven Hao, Zishuo Zhang, Xiaolin Chai, Judy Jiao, Zesong Li, Jian Wu, Kai Sun, Kun Jiang, et al. Pandaset: Advanced sensor suite dataset for autonomous driving. In *Proc. of the IEEE Intl. Conf. on Intelligent Transportation Systems (ITSC)*, pages 3095–3101, 2021.

[218] Zeqi Xiao, Wenwei Zhang, Tai Wang, Chen Change Loy, Dahua Lin, and Jiangmiao Pang. Position-guided point cloud panoptic segmentation transformer. *Intl. Journal of Computer Vision (IJCV)*, 133(1):275–290, 2025.

[219] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M. Alvarez, and Ping Luo. SegFormer: Simple and efficient design for semantic segmentation with transformers. In *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, volume 34, pages 12077–12090, 2021.

[220] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. SegFormer: Simple and efficient design for semantic segmentation with transformers. *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 34:12077–12090, 2021.

[221] Chenfeng Xu, Bichen Wu, Zining Wang, Wei Zhan, Peter Vajda, Kurt Keutzer, and Masayoshi Tomizuka. SqueezeSegV3: Spatially-adaptive convolution for efficient point-cloud segmentation. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 1–19, 2020.

[222] Jianyun Xu, Ruixiang Zhang, Jian Dou, Yushi Zhu, Jie Sun, and Shiliang Pu. RPVNet:a deep and efficient range-point-voxel fusion network for LiDAR point cloud segmentation. In *Proc. of the IEEE/CVF Intl. Conf. on Computer Vision (ICCV)*, pages 16024–16033, 2021.

[223] Xiang Xu, Lingdong Kong, Hui Shuai, and Qingshan Liu. FRNet: Frustum-range networks for scalable LiDAR segmentation. *arXiv preprint*, 2023.

[224] Xu Yan, Jiantao Gao, Chaoda Zheng, Chao Zheng, Ruimao Zhang, Shuguang Cui, and Zhen Li. 2DPass: 2D priors assisted semantic segmentation on LiDAR point clouds. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 677–695, 2022.

[225] Xu Yan, Chaoda Zheng, Zhen Li, Sheng Wang, and Shuguang Cui. PointASNL: Robust point clouds processing using nonlocal neural networks with adaptive sampling. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 5589–5598, 2020.

[226] Jie Yin, Ang Li, Tao Li, Wenxian Yu, and Danping Zou. M2DGR: A multi sensor and multi scenario SLAM dataset for ground robots. *IEEE Robotics and Automation Letters (RA-L)*, 7(2):2266–2273, 2021.

[227] Zhu Yixian, Ma Teng, Fan Jiajia, Jiang Yanqing, Li Ye, Liao Yulei, and Qi Chi. Robust underwater slam fusing bathymetric and range information. *Measurement*, 242:116223, 2025.

[228] Fei Yuan, Fengqi Xiao, Kaihan Zhang, Yifan Huang, and En Cheng. Noise reduction for sonar images by statistical analysis and fields of experts. *Journal of Visual Communication and Image Representation (JVCIR)*, 74, 2021.

[229] Mingxiao Yuan, Ye Li, Yueming Li, Shuo Pang, and Jucheng Zhang. A fast way of single beacon localization for AUVs. *Applied Ocean Research*, 119:103037, 2022.

[230] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 173–190, 2020.

[231] Yaohua Zha, Jinpeng Wang, Tao Dai, Bin Chen, Zhi Wang, and Shu-Tao Xia. Instance-aware dynamic prompt tuning for pre-trained point cloud models. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 14161–14170, 2023.

[232] Yaohua Zha, Yanzi Wang, Hang Guo, Jinpeng Wang, Tao Dai, Bin Chen, Zhihao Ouyang, Xue Yuerong, Ke Chen, and Shu-Tao Xia. PMA: Towards parameter-efficient point cloud understanding via point mamba adapter. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 16976–16986, 2025.

[233] Feihu Zhang, Jin Fang, Benjamin Wah, and Philip Torr. Deep FusionNet for point cloud semantic segmentation. In *Proc. of the Europ. Conf. on Computer Vision (ECCV)*, pages 644–663, 2020.

[234] Mabel M Zhang, Woen-Sug Choi, Jessica Herman, Duane Davis, Carson Vogt, Michael McCarrin, Yadunund Vijay, Dharini Dutia, William Lew, Steven Peters, et al. Dave aquatic virtual environment: Toward a general underwater robotics simulator. In *IEEE/OES Autonomous Underwater Vehicles (AUV) Symposium*, 2022.

[235] Wenyao Zhang, Shipeng Lyu, Chen Yao, Feng Xue, Zheng Zhu, and Zhenzhong Jia. Analysis of robot traversability over unstructured terrain using information fusion. In *Proc. of the Intl. Conf. on Advanced Robotics and Mechatronics (ICARM)*, pages 413–418, 2022.

[236] Yang Zhang, Zixiang Zhou, Philip David, Xiangyu Yue, Zerong Xi, Boqing Gong, and Hassan Foroosh. PolarNet: An improved grid representation for online lidar point clouds semantic segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9601–9610, 2020.

[237] Yizhe Zhang, Tao Zhou, Shuo Wang, Peixian Liang, Yejia Zhang, and Danny Z Chen. Input augmentation with SAM: Boosting medical image segmentation with segmentation foundation model. In *Proc. of the Intl. Conf. on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*, pages 129–139, 2023.

[238] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip HS Torr, and Vladlen Koltun. Point transformer. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 16259–16268, 2021.

[239] Yiming Zhao, Lin Bai, and Xinming Huang. FIDNet: LiDAR point cloud semantic segmentation with fully interpolation decoding. In *Proc. of the IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, pages 4453–4458, 2021.

[240] Xin Zhou, Dingkang Liang, Wei Xu, Xingkui Zhu, Yihan Xu, Zhikang Zou, and Xiang Bai. Dynamic adapter meets prompt tuning: Parameter-efficient transfer learning for point cloud analysis. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 14707–14717, 2024.

[241] Qinfeng Zhu, Lei Fan, and Ningxin Weng. Advancements in point cloud data augmentation for deep learning: A survey. *Pattern Recognition*, 153:110532, 2024.

[242] Xinge Zhu, Hui Zhou, Tai Wang, Fangzhou Hong, Yuexin Ma, Wei Li, Hongsheng Li, and Dahua Lin. Cylindrical and asymmetrical 3D convolution networks for LiDAR segmentation. In *Proc. of the IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, pages 9939–9948, 2021.

[243] Zeyu Zhu, Nan Li, Ruoyu Sun, Huijing Zhao, and Donghao Xu. Off-road autonomous vehicles traversability analysis and trajectory planning based on deep inverse reinforcement learning. *Proc. of the IEEE Vehicles Symposium (IV)*, pages 971–977, 2020.

[244] Xueyan Zou, Jianwei Yang, Hao Zhang, Feng Li, Linjie Li, Jianfeng Wang, Lijuan Wang, Jianfeng Gao, and Yong Jae Lee. Segment everything everywhere all at once. *Proc. of the Conf. on Neural Information Processing Systems (NeurIPS)*, 36:19769–19782, 2023.