infrastructures

MDPI

Article

Hierarchical SVM for Semantic Segmentation of 3D Point Clouds for Infrastructure Scenes

Mohamed Mansour *, Jan Martens and Jörg Blankenbach

Geodetic Institute and Chair for Computing in Civil Engineering & Geo Information Systems, RWTH Aachen University, Mies-van-der-Rohe-Str. 1, 52074 Aachen, Germany; jan.martens@gia.rwth-aachen.de (J.M.); blankenbach@gia.rwth-aachen.de (J.B.)

* Correspondence: mohamed.mansour@gia.rwth-aachen.de

Abstract: The incorporation of building information modeling (BIM) has brought about significant advancements in civil engineering, enhancing efficiency and sustainability across project life cycles. The utilization of advanced 3D point cloud technologies such as laser scanning extends the application of BIM, particularly in operations and maintenance, prompting the exploration of automated solutions for labor-intensive point cloud modeling. This paper presents a demonstration of supervised machine learning—specifically, a support vector machine—for the analysis and segmentation of 3D point clouds, which is a pivotal step in 3D modeling. The point cloud semantic segmentation workflow is extensively reviewed to encompass critical elements such as neighborhood selection, feature extraction, and feature selection, leading to the development of an optimized methodology for this process. Diverse strategies are implemented at each phase to enhance the overall workflow and ensure resilient results. The methodology is then evaluated using diverse datasets from infrastructure scenes of bridges and compared with state-of-the-art deep learning models. The findings highlight the effectiveness of supervised machine learning techniques at accurately segmenting 3D point clouds, outperforming deep learning models such as PointNet and PointNet++ with smaller training datasets. Through the implementation of advanced segmentation techniques, there is a partial reduction in the time required for 3D modeling of point clouds, thereby further enhancing the efficiency and effectiveness of the BIM process.

Keywords: building information modeling; machine learning; infrastructure; 3D laser scanning; semantic segmentation; support vector machine

Citation: Mansour, M.; Martens, J.; Blankenbach, J. Hierarchical SVM for Semantic Segmentation of 3D Point Clouds for Infrastructure Scenes. *Infrastructures* **2024**, *9*, 83. https://doi.org/10.3390/infrastructures9050083

Academic Editor: David Lattanzi

Received: 1 February 2024
Revised: 28 April 2024
Accepted: 1 May 2024
Published: 6 May 2024

1. Introduction

Building information modeling (BIM) is widely recognized in global government and civil engineering communities as a highly efficient tool for optimizing construction processes throughout a structure's life cycle [1,2]. It involves creating a digital model that serves as a comprehensive description of the structure's components. This model is collaboratively generated and continually updated at key stages of the project. Initially, during the planning and design phase, BIM relies on virtual models (project information models) to envision the structure. As the project progresses, these models evolve into as-built or as-is representations, reflecting the actual built condition and forming the basis of asset information models (AIM) for operation [3]. The as-is BIM model can be expanded by integrating real-time data from the built environment through Internet of Things (IoT) devices, thereby enhancing its functionality in project operations and maintenance while also laying the foundation for implementing a digital twin concept [4]. This innovative approach not only provides a detailed representation of the structure's geometric and semantic aspects but also accurately captures its position and height [5]. This approach addresses the longstanding challenge of creating a unified model that encompasses all aspects of complex infrastructure projects, offering a practical solution to this issue.

In the realm of bridge infrastructure, the imperative for effective maintenance, operation, and facility management stands paramount. Bridges, as vital components of transportation networks, undergo continuous wear and tear, necessitating proactive maintenance strategies to ensure safety, reliability, and longevity [6,7]. Traditional approaches to bridge maintenance and facility management often grapple with inefficiencies and challenges in data management, coordination, and decision-making processes [8,9]. In this context, the adoption of BIM technology emerges as a transformative solution to address these challenges. BIM offers a holistic approach to bridge management, providing a digital platform for comprehensive data integration, visualization, and analysis throughout the asset lifecycle. This enhances the ability to make data-informed maintenance decisions, mitigate risks, and allocate resources efficiently [10].

Construction projects typically fall into two main modeling scenarios: greenfield and brownfield. Greenfield involves generating new information about built assets from design to construction, ideally resulting in an as-built model identical to the as-planned model. However, deviations between the two models are common [11]. In contrast, brownfield scenarios require considering an existing asset's information, taking into account its nature and quality for the project. This scenario is especially relevant for older structures. While digital models of existing assets may be available in rare cases, updates are often necessary due to modifications, damage, and repairs [5].

In both greenfield and brownfield scenarios, the process of creating an as-built model typically involves high-resolution data capture using 3D laser scanning or photogrammetry [12]: commonly referred to as Scan-to-BIM (Scan2BIM). The captured dataset is preprocessed to eliminate redundancy and noise in order to ensure the quality of subsequent analyses. Following this, the data are segmented to identify regions of interest and are classified into object categories to facilitate more detailed analysis and interpretation through a process known as semantic segmentation. This semantic segmentaton step paves the way for the application of various geometric algorithms that play a crucial role in generating 3D geometric models [13–15]. The entire workflow is depicted in Figure 1, which illustrates the sequential progression from data capture to BIM model creation.

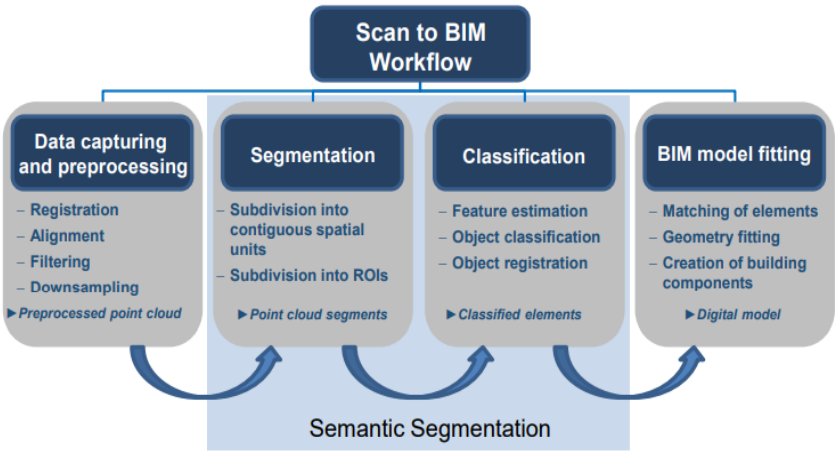


Figure 1. Steps of Scan2BIM showing where our approach fits in the automation of the semantic segmentation stage for 3D point clouds.

Scan2BIM involves manual labor, wherein users transform point clouds acquired from various reality capture devices into 3D models using a BIM software, like Autodesk Revit created by the american multinational software corporation [16]. This process is known for being time-consuming and intricate, especially when dealing with complex geometries [17]. The automation of this process typically happens in two steps: segmentation and classification of the input point cloud and 3D modeling of the primary asset elements [18].

Semantic segmentation of 3D point clouds for bridges encompasses various approaches, each offering unique advantages and challenges. The bottom-up approach

involves analyzing individual points and grouping them based on geometric or radiometric characteristics to identify semantic elements. This method relies on geometric features such as curvature, normal vectors, or intensity values to classify points into semantic categories [19–21]. However, this method lacks the ability to capture higher level features and is sensitive to noise and occlusions accompanied with the point cloud. In contrast, the top-down approach starts with a global understanding of the scene, such as structural components or architectural features, and then refines the segmentation at finer levels of detail. This method often involves predefined rules or templates based on domain knowledge to classify point clusters into semantic categories [22]. However, such an approach may lack adaptability to diverse bridge structures that deviate from conventional patterns. Additionally, learning-based approaches utilize machine learning algorithms to extract semantic labels from 3D point clouds and achieve accurate semantic segmentation [23–25]. Yet the deployment of these techniques for bridge semantic segmentation faces challenges due to the scarcity of labeled large-scale real bridge point cloud datasets for training models and the computational demands associated with such approaches.

This study aims to automate the Scan2BIM process, with a focus on 3D point cloud semantic segmentation to reduce manual labor and improve the efficiency of generating as-is BIM models. Leveraging learning-based approaches for high segmentation performance, our methodology addresses challenges hindering their implementation in infrastructure domains. Our automated method uses machine learning to achieve semantic segmentation of point clouds by integrating various strategies to enhance different workflow stages. Compared to deep learning models, the introduced classical machine learning approach demonstrates similar or better performance, with the advantage of requiring significantly less training data.

2. Related Work

Recent research has shown a growing interest in automating the Scan2BIM process [16]. Some studies have focused solely on automatically generating 3D geometric models of bridge datasets, overlooking the automation of semantic segmentation, which affect the quality of the generated 3D models [26–28]. Alternatively, other studies have approached semantic segmentation as a detection problem: detecting primitives and then labeling them semantically. For instance, Zhang et al. [19] introduced a method using sparsity-inducing optimization for identifying planar patches within noisy point clouds, but it is limited to planar surfaces and struggles with low-density regions such as pier patches. Dimitrov and Golparvar-Fard [20] proposed an improved region growing (RG) method for object detection in point clouds that is adept at handling curved surfaces but tends to oversegment objects in the presence of occlusions. Additionally, Xu et al. [21] presented a probabilistic segmentation model using octrees, but its segmentation accuracy is highly sensitive to variations in voxel size.

Other investigations have explored alternative techniques from image processing to automatically segment and generate BIM models from point cloud data. Martens et al. [29] introduced the VOX2BIM method, which converts point clouds into voxel representations and employs 2D representations and morphological operations for BIM model generation. While this approach offers computational advantages, it risks data loss through conversion, and their research primarily focused on indoor environments. Additionally, other research works have delved into automating the workflow for point cloud processing, which is an integral component of the Scan2BIM process [30]. Despite these efforts, these studies did not address the semantic segmentation process or BIM model generation. Another study conducted by Charles and Boehm [31] focused on automating the geometric generation of 3D BIM models for BIM applications utilizing RANSAC (RANdom SAmple Consensus) for segmenting walls, floors, and ceilings and then constructing IFC (Industry Foundation Class) geometry. However, semantic segmentation was constrained to planar surfaces within indoor environments.

Various terms are employed in scholarly discussions to characterize point cloud segmentation and classification, including semantic segmentation, instance segmentation, point labeling, and point-wise classification [32–34]. Semantic segmentation involves assigning semantic labels based on classes, while instance segmentation goes further to distinguish individual instances or objects within the same class, providing a more detailed understanding of the scene. For simplicity, the term "semantic segmentation" will be used in this work and includes the segmentation and classification steps in Figure 1.

Machine learning, including both classical and deep learning methods, plays a crucial role in training models for classification tasks. Classical machine learning relies on predefined features and statistical models, whereas deep learning autonomously extracts features from raw data [35]. In the context of 3D point cloud analysis, our approach involves neighborhood selection, feature extraction, feature selection, and point cloud classification, facilitating automatic semantic segmentation [36]. We discuss both classical and deep-learning-based segmentation methods, elucidating their respective processes, advantages, and limitations.

2.1. Classical Machine Learning Method

The classical machine-learning-based semantic segmentation process involves four key stages: neighborhood selection, feature extraction, feature selection, and supervised classification (Figure 2). To ensure a comprehensive understanding, we will succinctly discuss each stage and highlight techniques employed in the existing literature.

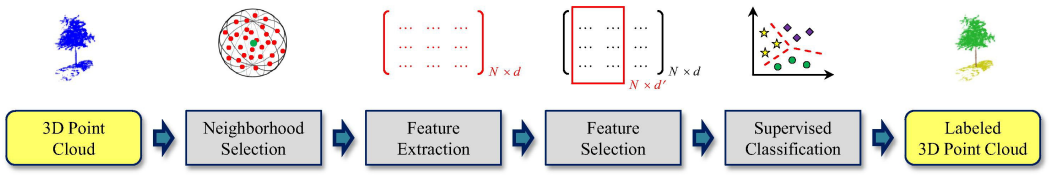


Figure 2. Workflow of point cloud semantic segmentation using supervised machine learning [37].

2.1.1. Neighborhood Selection

Defining relevant 3D point neighborhoods is crucial for 3D point cloud semantic segmentation, and various definitions exist, including spherical, cylindrical, and k-nearest neighborhoods [38–41]. Spherical neighborhoods encompass points within a predetermined radius forming a spherical shape around a central point. On the other hand, cylindrical neighborhoods involve considering 2D projections, typically resulting in a cylindrical shape around a central axis. K-nearest neighborhoods entail selecting a set number of nearest points based on their distance to a given point, forming a neighborhood based on proximity. While radius neighborhoods are conceptually appropriate, they may be less effective with varying point densities [42,43]. Studies have explored two approaches to select the neighborhood scale around a 3D point. The first employs a fixed scale parameter approach, like k or radius, which is uniformly applied to all points. However, this method may not sufficiently accommodate variations in size and can be influenced by fluctuations in point cloud densities. Additionally, it relies on heuristic or empirical knowledge to determine the fixed scale parameter [37]. On the other hand, the second method, known as the individual neighborhood approach, optimizes scale parameters independently for each 3D point and considers their local density and structure. This approach mitigates the limitations of the fixed scale method by employing eigenentropy and dimensionality-based techniques for scale parameter selection [37,44,45].

2.1.2. Feature Extraction

In point cloud semantic segmentation, defining a suitable representation for a 3D point X depends on the designated neighborhood. These neighborhoods possess distinct features, which can be extracted either at a single scale or across multiple scales. Employing

feature extraction at a single scale can be advantageous for computational efficiency, but it requires precise scale selection to adequately capture representative features. On the other hand, extracting features at multiple scales has proven to be more effective by providing a comprehensive representation of local 3D geometry [41,42,46,47]. However, this approach presents challenges such as determining scale selection and its distribution and computational complexities arising from variations in point densities. From a mathematical standpoint, the extracted features occupy distinct positions in a high-dimensional space and are represented as a feature vector. The research incorporates various feature types, including parametric features [48], sampled features [49], and metrical features [50–52], tailored to the specific task at hand.

2.1.3. Feature Selection

Following the extraction of diverse features, feature selection methods can be employed to delineate a subset of features from the input. This subset aims to efficiently capture the essential characteristics of the input data: mitigating noise or irrelevant features and ultimately yielding robust predictive results [53]. Utilizing feature selection techniques offers the opportunity to gain valuable insights, optimize computational efficiency, and enhance prediction accuracy. Various feature selection methods are available in the literature, with each being suitable for specific tasks [54].

2.1.4. Supervised Classification

In model training, three primary machine learning approaches are employed: supervised learning, unsupervised learning, and semi-supervised learning. In supervised learning, the classification model learns to categorize data using examples provided during training. It gives probabilities for its predictions and can be adjusted to avoid overfitting. Unsupervised learning focuses on discovering correlated features between data items without the need for labeled training data. It is applied in tasks like clustering, dimensionality reduction, and feature learning [55]. Semi-supervised learning strikes a balance between supervised and unsupervised approaches and uses labeled and unlabeled data to enhance learning accuracy, which is particularly useful when obtaining robust supervision information is challenging [56,57].

2.1.5. Semantic Segmentation

Supervised machine learning algorithms have shown reasonable results when dealing with point cloud semantic segmentation processes [37,58]. This method is also applied by Brodu et al. [42] to segment and classify complex natural scenes using features derived from different scales (Figure 3).

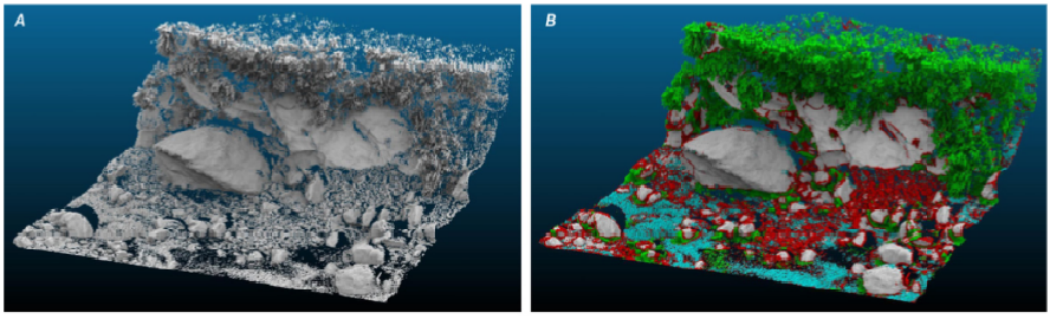


Figure 3. Result of the classification process for the mountain–river dataset: (**A**) original river scene and (**B**) classification (green: vegetation, gray: bedrock, red: gravel, and blue: water) [42].

Supervised machine learning algorithms, including support vector machine (SVM), random forest, k-nearest neighbor, and naive Bayes, find applications in 3D point cloud semantic segmentation. Among these, SVM [59] stands out for its proficiency in handling both linear and non-linear problems, with demonstrated high accuracy in identification and

classification tasks in computer vision and image processing [60,61]. In point cloud semantic segmentation, SVM has shown commendable performance, often in conjunction with other algorithms [37,58]. Osisanwo et al. [62] conducted a comprehensive investigation and favored SVM for its highest accuracy among compared algorithms. Through the kernel trick, input features are mapped to high-dimensional spaces, allowing for the identification of optimal hyperplanes that effectively separate samples from each category [63] (Figure 4). However, it is worth noting that SVMs can face challenges such as: algorithm complexity when dealing with large datasets, difficulty solving multiclass classification problems, and performance issues with imbalanced datasets [64].

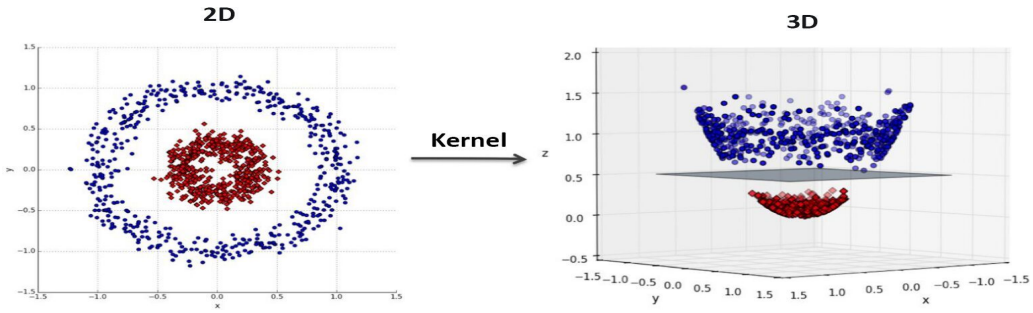


Figure 4. Using kernel function to map the dataset from 2D space to 3D space and constructing a separating surface between two classes, shown in red and blue [65].

For multi-class classification tasks in 3D point cloud analysis of infrastructure scenes, a specific approach is needed because of the involvement of multiple object classes such as vegetation, roads, and bridges. To address multi-class classification tasks using supervised binary classifiers designed for two-class discrimination, researchers employ a strategy known as "transformation to binary" [66]. Employing this strategy into an SVM involves breaking down the multi-class problem into binary classification problems using two main techniques. The "one vs. one" technique pairs each class against every other class, resulting in a total of k classifiers, where k is computed using Equation (1), and n represents the number of classes in the scene. Conversely, the "one vs. rest" technique compares each class against all others, resulting in a number of classifiers equal to the total number of classes. For multi-class classification tasks, an SVM has often been utilized alongside these techniques [67,68], and it can be integrated with decision tree architectures [69].

$$k = \frac{n(n-1)}{2} \tag{1}$$

2.2. Deep Learning Method

State-of-the-art point cloud semantic segmentation approaches are based on deep learning techniques [25,70–75]. The workflow of these techniques can be summarized into three main steps, as shown in Figure 5.

- Forward propagation: The input point cloud is fed into the deep learning model, and the model generates an output. During this step, the point cloud is transformed into a set of feature maps, which are then passed through multiple layers of the neural network to generate a final prediction.
- Backward propagation: The error, calculated from the loss function using the ground truth labels, is propagated back through the network to adjust the weights and biases of each layer in the model. This involves calculating the gradient of the loss function with respect to the weights and biases and using this information to update the values of these parameters.
- Optimization: An optimization algorithm, such as stochastic gradient descent (SGD) or Adam, is employed to adjust the weights and biases of the neural network based

on the gradients calculated in the backward propagation step. The objective is to find the values of the parameters that minimize the loss function.

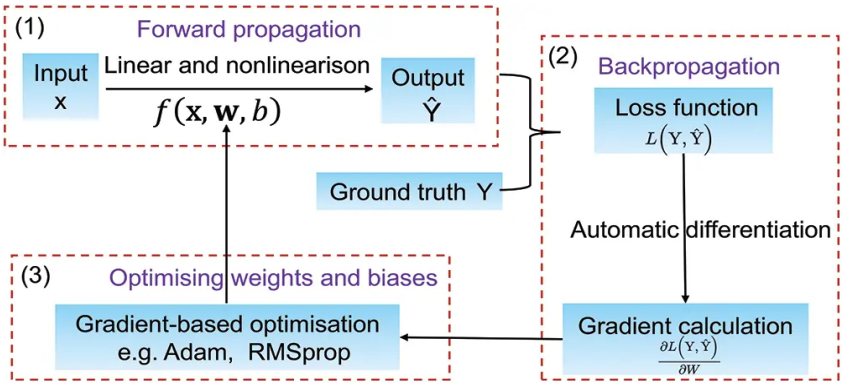


Figure 5. Deep learning workflow showing the three main steps [76].

Researchers commonly face the challenge of irregular points in point clouds and resort to transformations like 3D voxel grids or image collection before inputting them into deep learning architectures [77]. PointNet addresses this challenge by directly processing sets of points and utilizing multi-layer perceptrons and a symmetric function for effective handling of neighboring point information [24,78]. Despite its strengths, PointNet may struggle with detecting local features, potentially limiting its ability to generalize in complex scenes. PointNet++ builds upon this concept by recursively applying PointNet on partitioned input point clouds to capture local features at multiple scales [79]. Both architectures rely on the multi-layer perceptron (MLP) approach, which is widely used in the literature for classification and segmentation tasks.

Another noteworthy approach in deep learning is the utilization of convolutional neural networks (CNNs), which have demonstrated state-of-the-art performance across a wide range of computer vision tasks. In the context of 3D point cloud semantic segmentation, Thomas et al. [25] introduced a KPConv model that represents a significant advancement. This method leverages the power of CNNs to enhance the accuracy and efficiency of segmenting 3D point clouds, making it a pivotal tool in this domain.

2.3. Advantages and Limitations

In point cloud semantic segmentation, Weinmann et al. compared classical machine learning algorithms to deep learning models and showed that classical methods may outperform deep learning models in terms of characteristic feature utilization [37]. Deep learning approaches, while powerful, come with challenges like reliance on large datasets, substantial computational resources needed for training, and difficulties with interpreting deep networks [80]. These challenges can be addressed by employing classical machine learning methods. For instance, Zhang et al. introduced PointHop, which is a machine learning approach tailored for point cloud semantic segmentation [81]. PointHop exhibited significantly reduced training times compared to deep-learning-based methods and presents a promising alternative.

In the domain of autonomous driving, an SVM was assessed alongside a recurrent neural network (RNN) and a feed forward neural network (FFNN) to analyze human lane change behavior in a simulated environment [82]. The study compared the three techniques using various feature combinations and showed that SVM produced the most effective results and identified the most suitable feature combinations.

2.4. Research Gaps

Despite advancements in automating the semantic segmentation of the Scan2BIM process, several research gaps persist. These include:

- The absence of utilizing learning-based methods for the semantic segmentation of 3D point clouds of bridges;
- Uncertainty regarding the application of learning-based methods for semantic segmentation of bridge point clouds due to limitations in training datasets and computational time;
- Limited exploration into optimizing classical machine learning approaches for the semantic segmentation of 3D point clouds;
- Lack of a detailed comparison between classical machine learning and deep learning approaches in 3D point cloud semantic segmentation.

This paper aims to bridge these research gaps by investigating various techniques applicable at each stage of the semantic segmentation workflow for bridge datasets using classical machine learning. Additionally, it provides a comparison with state-of-the-art deep learning approaches.

3. Methodology

In this work, we semantically segment 3D point clouds of bridges based on classical supervised machine learning, for which a training dataset is required. For supervised learning approaches, standard supervised classifiers are applied, such as support vector machine [67] or random forest [52]. After illustrating SVM's effectiveness compared to other models across multiple application fields in previous sections, it appears plausible that SVM could be similarly beneficial for infrastructure overall. Given these considerations, this work will delve into the utilization of the support vector machine algorithm to develop a hierarchical support vector machine (HSVM) methodology for the semantic segmentation of 3D point clouds of infrastructure assets.

Figure 6 illustrates our workflow and showcases the developed semantic segmentation method, which is achieved by combining various techniques.

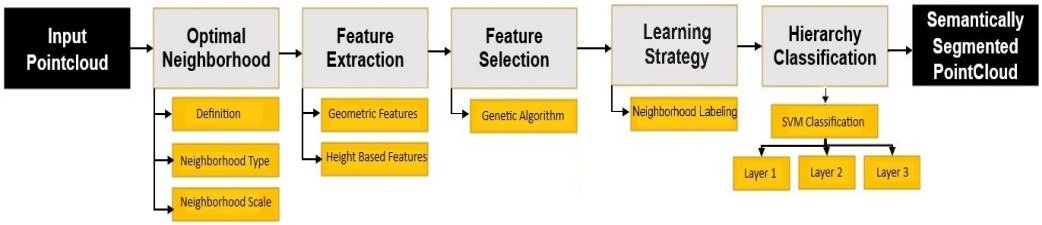


Figure 6. Workflow of our methodology showing the integration of different strategies (optimal neighborhood definition, geometric feature extraction, optimizing the feature selection, and hierarchy classification using SVM).

3.1. Optimal Neighborhood

This section outlines our method for determining the neighborhood of a 3D point in the point cloud semantic segmentation workflow using classical machine learning approaches. Addressing the challenge of randomly selecting the scale parameter r, which is based on empirical scene knowledge, we optimize the selection of this parameter utilizing the local 3D structure of a 3D point neighborhood. This integration of neighborhood parameter selection with its local 3D structure ensures that each point's neighborhood is the most representative or optimal for that particular point.

3.1.1. Definition

The concept of an optimal neighborhood involves identifying a spatially proximal cluster of 3D points that share the same category as a designated point of interest. The process revolves around addressing two key questions: (1) determining the suitable neighborhood type and (2) selecting the appropriate scale. Subsequent sections will delve into each question to provide a comprehensive understanding of the process and reveal the challenges involved in establishing an optimal neighborhood in 3D spatial contexts.

3.1.2. Neighborhood Type

In addressing the first question concerning the selection of the neighborhood type and to better comprehend the type of neighborhood employed in this study, it is crucial to delineate the fundamental differences between commonly used approaches. Unlike spherical and cylindrical neighborhoods, k-nearest neighborhoods do not have fixed spatial dimensions in space. Instead, they always contain a fixed number of points, even if we subsample the data, as shown in Figure 7. Conversely, radius neighborhood definitions correspond to specific spatial regions and allow for a more consistent interpretation of geometrical features. However, this neighborhood type may become impractical in cases of significant point density variations, which is a challenge that can be addressed by proportionally subsampling the point cloud to match the neighborhood scale [43]. Therefore, we will utilize a combination of a spherical neighborhood type and adaptive subsampling to mitigate the drawbacks associated with this neighborhood type.

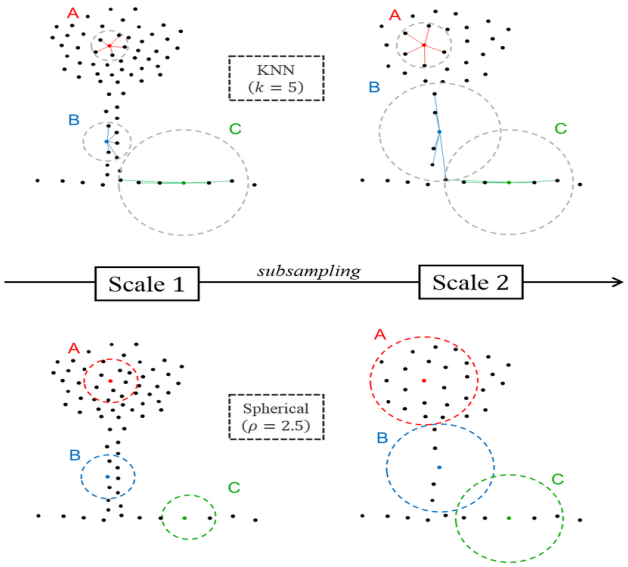


Figure 7. Behavior of the k-nearest neighborhoods and spherical neighborhoods under two different scales (Scale 1 and Scale 2) at three different locations in the scene (A, B, and C) [36].

Under the Euclidean norm, the spherical neighborhood $\mathcal{N}_i^r$ of point p_i at scale r is defined as the set of points p_k verifying:

$$p_k \in \mathcal{N}_i^r \Leftrightarrow \|p_i - p_k\| \leq r \tag{2}$$

3.1.3. Neighborhood Scale

To address the second question concerning the selection of the appropriate scale for the optimal neighborhood, we must implement an optimization workflow to define this scale accurately. However, before delving into this optimization process, it is beneficial to examine the traditional method commonly employed in many studies for determining this scale. Typically, the choice of the radius for the 3D point neighborhood is heuristic or empirical and is customized to the characteristics of each individual dataset. This implies that determining the suitability of a neighborhood based on its geometric features is deferred until after obtaining classification results. As depicted in Figure 6, neighborhood selection, feature extraction, and feature selection occur before classification, meaning that the suitability of the scale defined for the neighborhood will impact the extracted features and, consequently, will affect the segmentation workflow's performance. To address this challenge, the individual neighborhood selection approach, discussed in Section 2.1.1, presents a generic method for determining the neighborhood scale. This approach offers an alternative to alleviate the limitations of fixed-scale neighborhoods

in order to enable dataset generalization and to facilitate the definition of an optimal neighborhood. Importantly, in this approach, the neighborhood scale is directly linked to the features before classification occurs.

In the optimization process, it is crucial to establish a criterion to assess the appropriateness of a specific radius for a point neighborhood. The approach presented in this study seeks to automatically identify the optimal neighborhood radius using the entropy feature (E_f), which characterizes data distribution and provides insight into its structure. Entropy and data volume are closely linked: larger datasets exhibit greater disorder and entropy, whereas smaller datasets offer fewer choices, resulting in reduced entropy [83]. The aim here is to utilize this feature within a neighborhood to identify the optimal radius that enhances differentiation among the three primary geometric patterns (linearity, planarity, and scatter). This involves minimizing the entropy feature, thereby reducing the number of choices available.

Shannon entropy, a variant of entropy widely that is employed in information theory, physics, and continuous probability distributions, finds application in various optimization tasks [84,85]. In our context, Shannon entropy serves to analyze the distribution of three geometric behaviors within a group of points, thereby defining an energy function crucial for determining the optimal neighborhood radius. The energy function, as depicted in Equation (3), quantifies the uncertainty within a point neighborhood ($\mathcal{N}_{p_i}^r$) by considering the probabilities associated with encountering different geometric behaviors: a_{1D}, a_{2D}, and a_{3D} representing linearity, planarity, and scatter, respectively. Minimizing this function results in reduced uncertainty and assists with identifying the optimal radius that effectively captures the underlying structure of the point cloud.

$$E_f(\mathcal{N}_{p_i}^r) = -a_{1D}\ln(a_{1D}) - a_{2D}\ln(a_{2D}) - a_{3D}\ln(a_{3D}) \tag{3}$$

$$r_{E_f}^* = \arg\min E_f(N_{p_i}^r) \tag{4}$$

$$\text{where } r \in [r_{min}, r_{max}]$$

A lower $E_f(N_{p_i}^r)$ value in Equation (3) indicates stronger dominance of one dimensionality over the others, aiding with the selection of an optimal radius $r_{E_f}^*$ within the range $[r_{min}, r_{max}]$ to achieve a characteristic geometric feature.

The distribution probabilities in Equation (3), (a_{1D}, a_{2D}, and a_{3D}) are computed from the covariance matrix, which is often referred to as the 3D structure tensor and is derived from the 3D coordinates of the point neighborhood. The covariance matrix represents the relationships between different features of the 3D points and provides information about how they vary together. It can be defined as

$$S = \frac{1}{|N_i|+1} \sum_{p_j \in \{N_i, p_i\}} (p_j - \bar{p})(p_j - \bar{p})^T \tag{5}$$

where S is the structure tensor, $\bar{p}$ denotes the center of gravity of $\{N_i, p_i\}$, and N_i is the spherical neighborhood around reference point p_i.

Principal component analysis (PCA) is a statistical method employed to reduce the dimensionality of data while retaining most of its variability. When applied to the covariance matrix of 3D point neighborhoods, PCA reveals the primary axes of variation in the spatial arrangement of points. These axes are represented by the eigenvectors of the matrix, with each eigenvector denoting a direction of maximum variance. The corresponding eigenvalues signify the magnitude of variance captured by each eigenvector and enable us to gauge the extent of spread of the point cloud along different axes. Consequently, these eigenvalues serve as a basis for deriving various shape descriptors, such as a_{1D}, a_{2D}, and a_{3D}. Denoting the eigenvalues of the covariance matrix by $\lambda_{1,i}$, $\lambda_{2,i}$, $\lambda_{3,i} \in \mathbb{R}$, where $\lambda_{1,i} \geq \lambda_{2,i} \geq \lambda_{3,i} \geq 0$, the three geometric features (a_{1D}, a_{2D}, and a_{3D}) can be calculated as follows:

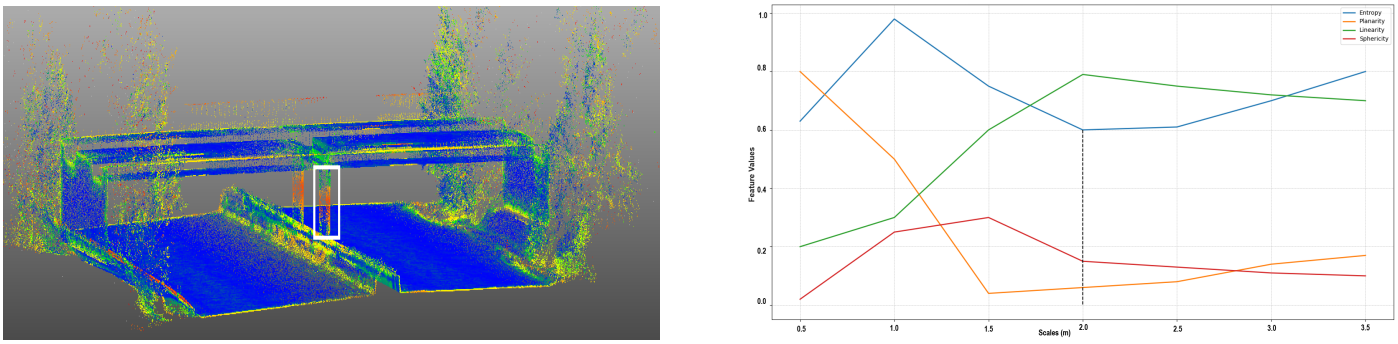
$$a_{1D} = \frac{\lambda_1 - \lambda_2}{\lambda_1} \tag{6}$$

$$a_{2D} = \frac{\lambda_2 - \lambda_3}{\lambda_1} \tag{7}$$

$$a_{3D} = \frac{\lambda_3}{\lambda_1} \tag{8}$$

Defining the search space for the optimal radius, which is determined by the selection of radius boundaries $(r_{\min}, r_{\max})$, presents a challenge due to its reliance on diverse data characteristics. Demantké et al. delved into the TLS and MMS datasets of heritage architecture buildings and delineate the $[r_{\min}, r_{\max}]$ space with 16 values [45]. While the upper boundary was determined based on the largest object (3 m), the lower boundary was chosen to ensure a minimum of 10 points at the smaller scale. To account for the characteristics of these datasets, the search radii were increased by a squared factor, thereby favoring the definition of optimal neighborhoods at smaller scales to capture finer architectural details. Adaptations should be made to suit infrastructure dataset characteristics and address the computational constraints when defining the search space.

In infrastructure environments, it is essential to identify key characteristics of specific objects, such as piers, at larger scales, while capturing finer details of others, like vegetation, at smaller scales. As illustrated in Figure 8, bridge piers typically exhibit more linearity within a scale range of approximately r = 1.2–3.5 m, with the lowest energy function value observed at r = 2.0 m, whereas smaller scales emphasize planar features. Achieving a balance between these factors is crucial when defining the search space to accommodate diverse object types.



(**a**) (**b**)

Figure 8. Finding the optimal neighborhood around the pier of the bridge, which is located at the radius with the minimum entropy value. (**a**) Linearity values for each point in the point cloud obtained by the optimal neighborhood search showing high linearity around the pier of the bridge (white box). (**b**) Finding the radius of the neighborhood corresponding to the minimum entropy value, showing more linearity over the other two features for the pier element.

To effectively capture features at larger scales, we incrementally increase the scale values r_s in the search space linearly using a constant Δr, thereby favoring larger radii for more frequent sampling. This adjustment aligns the r_s values with the areas of interest as defined in Equation (9), with the upper limit capped at 3.5 m, corresponding to the height of the bridge (Figure 8b). However, the linear incrementation of r_s values also entails a proportional rise in computational demands. This underscores the critical role played by both point density and selected neighborhood scales for delineating the search space for the optimal radius, given their close interdependence. While expanding the search radius

in dense regions escalates computational demands, reducing the radius may compromise the representation of distinct neighborhood features due to sparse point coverage.

$$r_s = r_{s-1} + \Delta r \tag{9}$$

where r_s is the radius at each scale s, and Δr is the change between two consecutive scales.

To address the computational challenges caused by larger scales, it is important to effectively manage the number of points at each scale. This helps capture strong global shape features in a neighborhood, with the understanding that not every point is essential for this task. To implement this strategic approach, we propose employing an adaptive subsampling process on the input point cloud, wherein the point cloud density decreases with an increase in the search radius. This approach achieves a balance by efficiently capturing fine details at smaller scales, which demand a higher point density, such as for vegetation, while also encompassing the broader characteristics of substantial objects at larger scales. Moreover, it addresses the challenges previously outlined for establishing the lower boundary of the search space. Equation (10) outlines the parameters for the adaptive subsampling method.

$$\rho_s = \rho_{s-1} * \Delta\rho \tag{10}$$

where ρ_s is the point cloud density at each scale s, and $\Delta\rho$ is the factor by which the density decreases and is set to 0.5.

To enrich the training dataset during model training, feature data from the subsampled point cloud can be projected onto the original dataset. This approach offers advantages, particularly in situations where computational efficiency is vital due to large dataset sizes. During the projection process, a neighborhood search is conducted to preserve spatial relationships between points. This entails identifying nearest neighbors in the subset for each point in the original point cloud, which establishes correspondence between points in the original and subset datasets.

The methodology proposed for defining the optimal neighborhood for 3D point clouds of infrastructure scenes while addressing computational demands and the scarcity of training datasets after subsampling will be evaluated through real-case experiments in the experimentation section.

3.2. Feature Extraction

Once the neighborhood around each 3D point is determined, the features within this neighborhood are gathered and combined to form a feature vector in the feature space. This study focuses on examining metrical features: each of which is characterized by a distinct value indicating a specific property. Specifically, two types of features are extracted and employed at various stages of the semantic segmentation process. The first type encompasses geometric features, often referred to as 3D eigenvalue-based features, while the second type pertains to height-based features.

3.2.1. Geometric Features

Using the eigenvalues obtained from the 3D structure tensor, as mentioned previously, different geometric features are extracted to describe the spatial distribution of points in the neighborhood N_i of a 3D point P_i. In our study, a diverse range of geometric features, including planarity, linearity, and sphericity features, combined with the x, y, and z coordinates of each point were found to be sufficient for delivering good results.

$$\text{planarity}: \quad P_\lambda = \frac{\lambda_2 - \lambda_3}{\lambda_1} \tag{11}$$

$$\text{linearity}: \quad L_\lambda = \frac{\lambda_1 - \lambda_2}{\lambda_1} \tag{12}$$

$$\text{sphericity}: \quad S_\lambda = \frac{\lambda_3}{\lambda_1} \tag{13}$$

In order to describe the verticality of the objects in the scene, the angle between the normalized normal vector and an "up"-vector (0,0,1) is calculated.

$$V = 1 - n_z \tag{14}$$

where n_z refers to the third component of the normal vector.

3.2.2. Height-Based Features

Since the scene contains a variety of vertical objects, it is very helpful to define different height features. For each 3D point P_i, the following height features are extracted:

- Maximal height features between any two points in the neighborhood N_i:

$$H_d = H_{max} - H_{min} \tag{15}$$

- Using the statistical moments to give a robust height feature. The standard deviation of the height is calculated using the following equation:

$$HSTD = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}(H_i - \bar{H})^2} \tag{16}$$

$$\bar{H} = \frac{1}{n}\sum_{i=1}^{n} H_i \tag{17}$$

3.3. Feature Selection

Each feature represents a specific measurable property that characterizes a particular geometric aspect. Employing an automated methodology, we identify effective combinations of these features to construct diverse feature sets for semantic segmentation of 3D points within our workflow. In this paper, we utilize a heuristic search algorithm—specifically, we employ the genetic algorithm (GA) [86]—to formulate an optimization problem for feature selection.

Incorporating the defined features into the genetic algorithm workflow involves encoding each feature into a binary value (0 or 1). A set of features can then be represented as a chromosome or string of binaries with a length equal to the number of features under investigation. In the optimized chromosome, a feature represented by a value of 0 indicates that it is either unnecessary for this specific feature set or contributes minimally to improving the results. Figure 9 illustrates the process of transforming a feature set into binary representations.

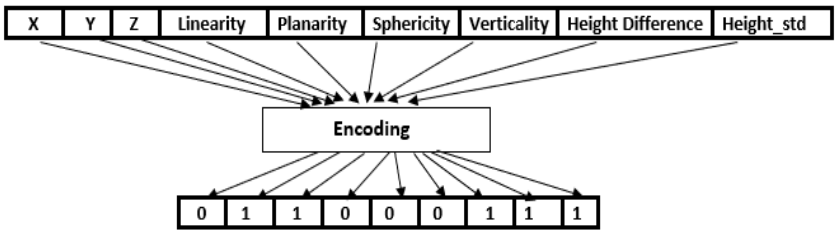


Figure 9. Converting a set of features to a chromosome involves assigning each feature a binary value.

The genetic algorithm workflow, depicted in Figure 10, for selecting the optimal feature set involves three primary steps: selection, cross-over, and mutation. In the selection step, two feature sets, referred to as parents in the genetic algorithm (GA), are chosen from the initial population based on their fitness values. We employ the tournament selection method [87], wherein three random feature sets are chosen from the initial population and

the best feature set is selected as the first parent based on its objective value. This process is repeated twice to select the second parent. During the cross-over step, features between the parents are altered to generate new children (or feature sets). Although various cross-over operators exist, we utilize the two-point cross-over method to increase the ability to explore the search space. This method involves interchanging binary values between two points, as depicted by the green middle segment in Figure 10. Subsequently, these new feature sets undergo mutation to create the new population for subsequent iterations. Finally, we monitor the objective value at each generation as well as the objective value at the last generation to obtain the lesser one in case convergence is not achieved in the end.

$$\text{Fitness Value} = 1 - \text{Classifier Accuracy} \tag{18}$$

The fitness value presented in Equation (18) reflects the classifier error within the HSVM model. Our objective is to minimize this error using the genetic algorithm.

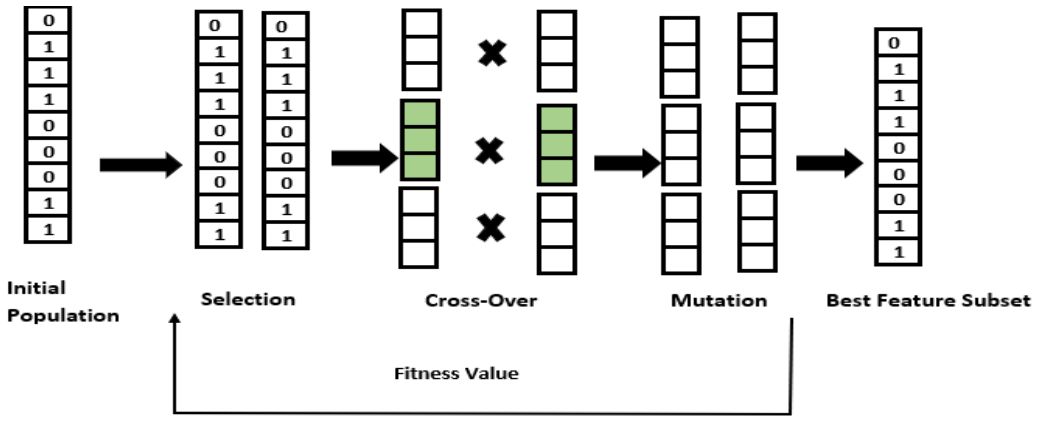


Figure 10. Workflow of the GA to define the best feature sets at each layer in the HSVM model.

3.4. Learning Strategy

During the training phase, the model utilizes a feature vector comprising geometric and height features for training. Two key considerations arise during this process: Firstly, ensuring even distribution of the training dataset enhances model generalization and prevents overfitting. Secondly, establishing the optimal strategy for labeling the neighborhood provided to the model during training is crucial.

Addressing the first aspect involves dealing with imbalanced datasets, for which several methods can be employed. In this study, we use techniques like resampling the dataset to make sure each class has the same number of samples. This is done by duplicating instances from the underrepresented class. Additionally, appropriate performance metrics are employed to evaluate the model's quality.

The prediction process yields four possible outcomes based on the consistency between the predicted label and the true label of the test sample. These outcomes are classified into true positive (*TP*), true negative (*TN*), false negative (*FN*), and false positive (*FP*). These outcomes serve as the basis for defining the following metrics:

(i) Precision (measure of correctness)

$$\text{precision} = \frac{TP}{TP + FP}; \tag{19}$$

(ii) Recall (measure of completeness)

$$\text{recall} = \frac{TP}{TP + FN}; \tag{20}$$

(iii) Accuracy (measures overall performance of the model)

$$\text{accuracy} = \frac{TP + TN}{TP + FN + FP + TN}; \tag{21}$$

(iv) Intersection over union (IoU)

$$\text{IoU} = \frac{TP}{TP + FN + FP}; \tag{22}$$

(v) Mean of IoU (mIoU)

$$\text{mIoU} = \frac{1}{C} \sum_{c \in C} IoU_c; \tag{23}$$

where C is number of classes.

Dealing with the second aspect of labeling the neighborhood involves two approaches: labeling based on the search point and labeling based on the majority of points within the neighborhood. We will investigate and assess both strategies using real datasets in the experimentation section.

3.5. Hierarchy Support Vector Machine Approach

Using the two techniques mentioned in Section 2.1.5, one vs. one and one vs. rest, the whole scene can be segmented using the HSVM approach. In this approach, the 3D point cloud is semantically segmented through three layers (see Figure 11). Each layer employs various SVM kernels and techniques to separate different features and categories within the dataset. In the first layer, the scene is classified into two categories: namely, vegetation and manmade objects, using the one vs. one technique. In the second layer, the classifier differentiates between ground and above-ground objects using the same technique as in the previous layer. Lastly, the 3D point cloud of the above-ground objects is semantically segmented into different segments such as abutment, footing, girder, railing, and pier. In this layer, the SVM classifier is applied using the one vs. rest technique.

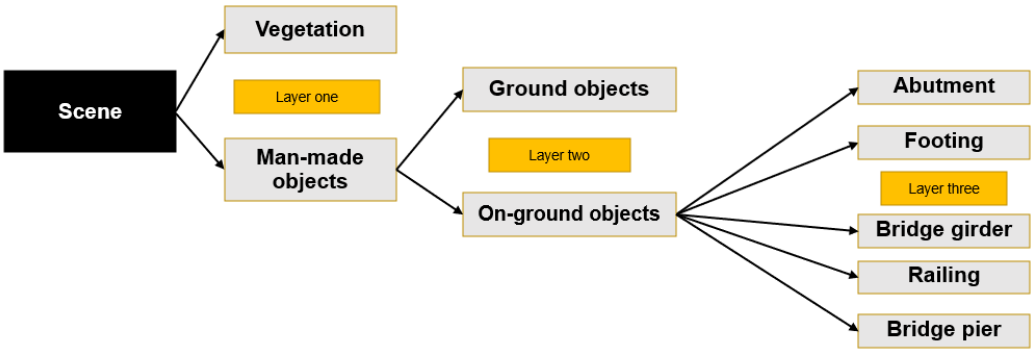


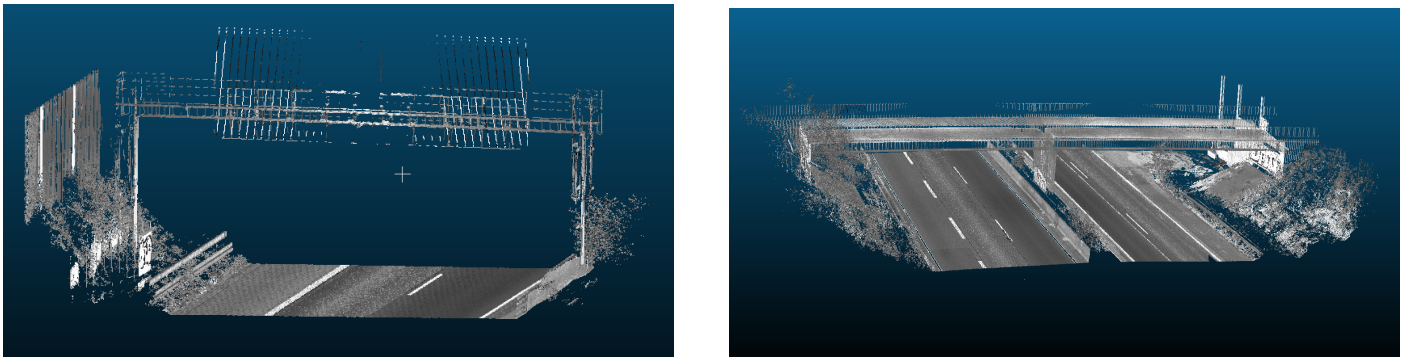
Figure 11. The hierarchy classification approach combining SVM solutions and the decision tree framework.

4. Evaluation Datasets

To assess the effectiveness of the HSVM methodology and compare it to the deep learning methods, we utilize several real-world datasets of infrastructure objects. These datasets contain different types of scenes from Straßen NRW Nord in Germany and German highways. One type of scene displays roads without bridge elements and the second type of scene illustrates roads with bridge elements (see Figures 12a and 12b, respectively). All scenes were captured using MLS.

In the HSVM model training phase, each class was represented by 20,000 points, resulting in 100,000 points for Scene 1 and 140,000 points for Scene 2. When testing the model, Scene 1 involved a total of 209,400 points, while Scene 2 utilized 149,114 points. Importantly, the test datasets were completely separate from the training datasets. The ground truth

labels for these scenes include a total of nine classes that cover all infrastructure buildings in the respective scenes. Table 1 provides a summary of the classes and the number of points used in both the training and testing phases.



(**a**) (**b**)

Figure 12. Scenes of the datasets used to evaluate the performance of the investigated models. (**a**) Dataset, Scene 1: noise barriers, road, vegetation, railing, and signs in the scene of Straßen NRW, with laser scan intensities shown in gray-scale. (**b**) Dataset, Scene 2: bridge elements, vegetation, railing, and road in the scene of Straßen NRW, with laser scan intensities shown in gray-scale.

Table 1. Summary of the data points used in the training and testing of the HSVM model.

Dataset	Classes	Training (pts)	Test (pts)
Scene 1	vegetation, road, railing, noise barrier, signs	100,000	209,400
Scene 2	vegetation, road, railing, abutment, girder, pier, footing	140,000	149,114

The evaluation of the PointNet, PointNet++, and KPConv models was performed using 10 datasets comprising the objects in Scene 1. It is noteworthy that these models utilized a significantly greater number of points for both training and evaluation than the HSVM model. Specifically, we employed a training set consisting of 61,995,915 points sourced from 58 distinct datasets, while the test set comprised 11,896,152 points. As part of the training process, the model underwent evaluation on two specific datasets encompassing a total of 2,186,724 points (see Table 2).

Table 2. Summary of data points used for the training, validation, and testing of the deep learning models.

Dataset	Classes	Training (pts)	Validation (pts)	Test (pts)
Scene 1	vegetation, road, railing, noise barrier, and signs	61,995,915	2,186,724	11,896,152

5. Implementation Details/Tools

The 3D point cloud semantic segmentation process described here is implemented in Python, with the preprocessing step that computes the optimal neighborhood for the point cloud performed in C++. Scikit-learn [88] provides a range of tools for adjusting SVM classifier parameters and offers various metrics for evaluating model performance. Matplotlib [89] offers different options for visualizing the results and provides an overview of the distribution of the classes in the dataset. The whole implementation of the HSVM method was run on a CPU, which is an important factor to be considered for the comparison between models. For the implementation of PointNet, PointNet++, and KPConv, the PyTorch library was used [90]. All experiments were conducted on the same machine with an Intel Core i9-12900K @3.20GHz CPU and an NVIDIA RTXA5000 GPU.

6. Experiments

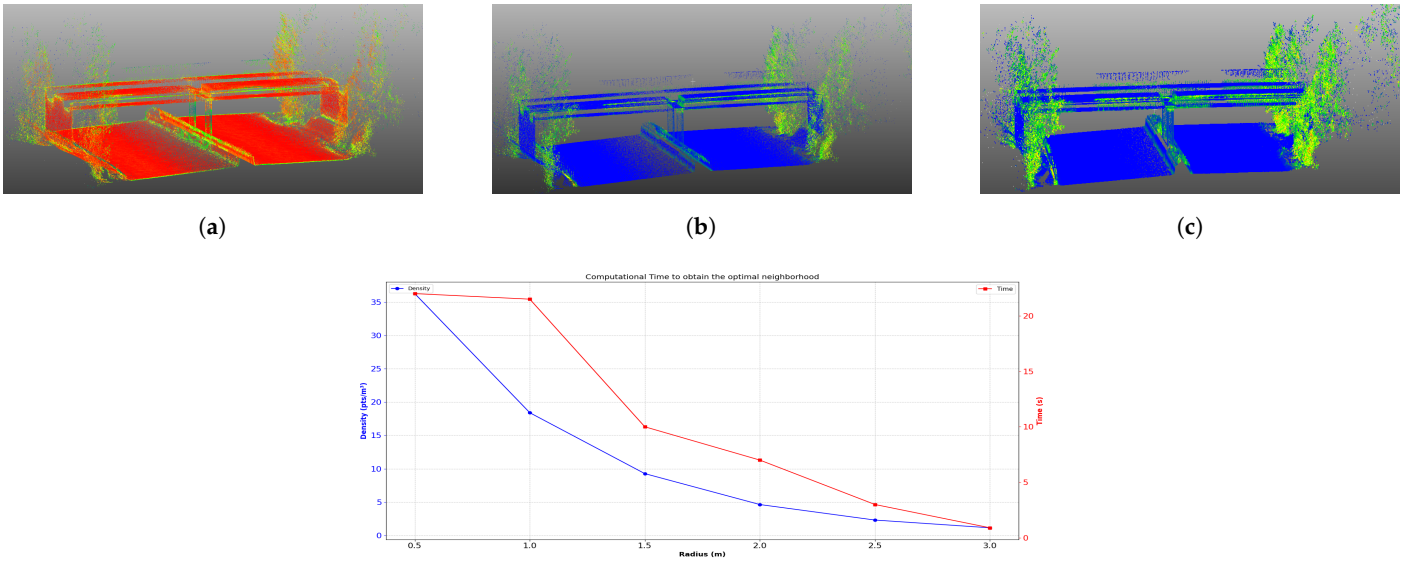
In the first part of this chapter, we evaluate our methodology, which we introduced previously, using real-world datasets of infrastructure scenes. In the second part, we experiment with various deep learning methods, including PointNet, PointNet++, and KPConv, to offer a comparative analysis of their performance.

6.1. Experiment 1: Our Methodology

We test each stage in the semantic segmentation workflow and demonstrate its overall impact on the performance of the HSVM model.

6.1.1. Optimal Neighborhood Estimation

With the methodology for identifying optimal neighborhoods established (Section 3.1), we proceed to assess its performance through comprehensive experiments conducted on a 3D point cloud dataset. Figure 13 provides a summary of the various aspects of the employed method.



(**a**) (**b**) (**c**) (**d**)

Figure 13. Estimating the optimal neighborhood to define robust features projected onto the original point cloud while utilizing adaptive subsampling for computational efficiency. (**a**) Planarity features from optimal neighborhood. (**b**) Sphericity features from optimal neighborhood. (**c**) Sphericity features after projection to the original point cloud. (**d**) Computational time required at each scale to define a neighborhood when employing adaptive subsampling with $\rho_{0.5}$ = 36.3 pts/m^3.

Figure 13a,b show the planarity and sphericity features extracted from the optimal neighborhood. Figure 13c demonstrates the expansion of the training dataset by projecting the sphericity features from the subsampled point cloud (1,000,000 pts) to the original point cloud (2,187,299 pts) within 45.1 s. Figure 13d illustrates the adaptive subsampling process applied to the initial point cloud, which had a density of $\rho_{0.5}$ = 36.3 pts/m^3. As the neighborhood search radius increases, further subsampling of the point cloud occurs. The corresponding computational time required at each scale to obtain the spherical neighborhood summed to a total time of 65 s for all scales.

6.1.2. Feature Selection Optimization

Employing the genetic algorithm to construct diverse feature sets for our HSVM model results in the identification of the optimal feature set at each layer. Table 3 summarizes the parameters of the genetic algorithm and the number of splits for the datasets. These specified parameters achieve a balance between computational time and accuracy.

Table 3. Parameters of the GA algorithm.

Parameters	Values
generations	10
population	20
cross-over probability	1
mutation probability	0.2
k-fold	5

The outcomes of the automated feature selection process and the type of the kernel used within each layer in the HSVM model to separate between the features in the feature space are shown in Table 4.

Table 4. Optimal feature set at each layer selected by the GA algorithm for the first dataset.

Best Chromosome	Decoded Feature Set	Fitness_VALUE	HSVM Layers	SVM Kernel
[1,1,1,1,1,1,1,0,0]	x,y,z, linearity, planarity, sphericity, verticality	0.026	layer 1	Linear
[1,1,1,1,0,0,1,1,0]	x,y,z, linearity, verticality, height difference	0.0126	layer 2	Linear
[1,1,1,0,0,0,1,1,0]	x,y,z, verticality, height difference	0.006	layer 3	RBF

Figure 14 illustrates the evolution of fitness values for the classifiers employed in each layer of the HSVM model across all generations. During optimization, the GA algorithm seeks the optimal feature set corresponding to the lowest fitness value, known as the global minima. The algorithm converged in the last generation for the first and second layers, defining the best feature set. However, convergence occurred early during Generation 2 for the third layer.

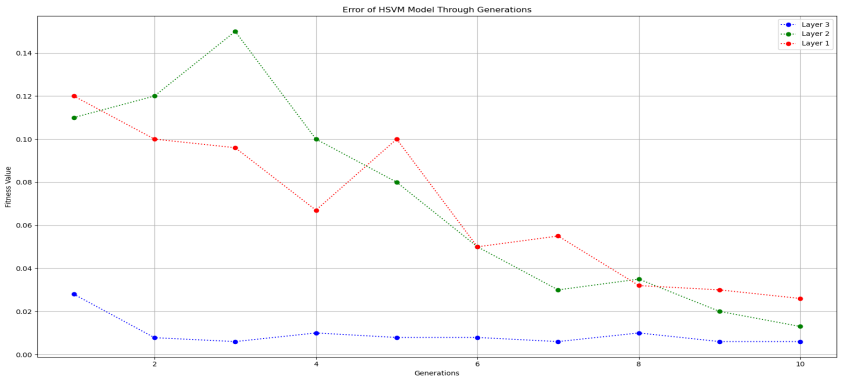


Figure 14. Minimizing errors for each classifier within the HSVM model across all generations.

6.1.3. Neighborhood Labeling

To explore various learning strategies in HSVM training, we utilized the dataset featuring bridge elements from Scene 2. Figure 15 presents results from both training methods in each layer and indicates labeling accuracies of 90.27% and 82.13% based on majority points and reference points in the neighborhood, respectively.

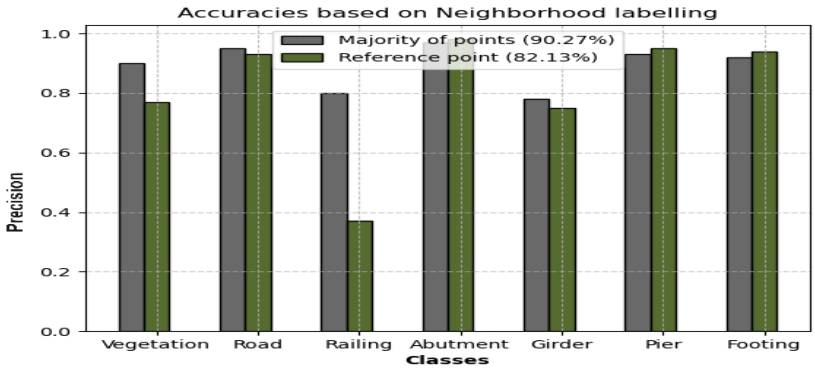
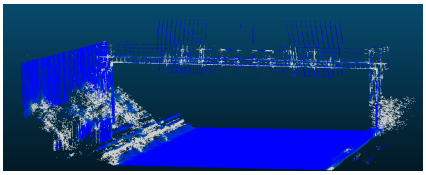


Figure 15. Comparison between both learning strategies employed in the training process of the HSVM approach for the dataset of Scene 2.

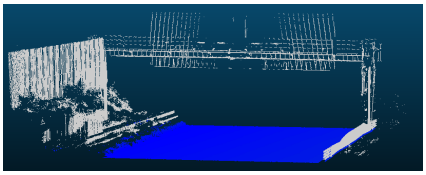
6.1.4. Hierarchical Support Vector Machine Approach

Our developed approach, as previously detailed in Section 3.5, involves three classification layers and will be tested on various datasets, including those with and without bridge elements.

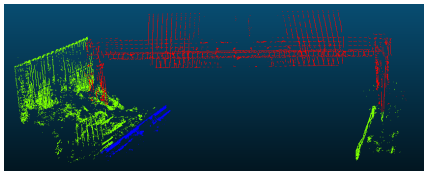
The HSVM model attained an overall accuracy of 91.94% in Scene 1 of the dataset, which captures a highway and encompasses five distinct classes. Figure 16 illustrates the model's performance at each classification layer, while Table 5 provides a detailed breakdown of classification performance across various classes. In Layer 1 (Figure 16a), the focus lies on distinguishing between vegetation and man-made objects, while Layer 2 (Figure 16b) separates ground and above-ground elements. Layer 3 (Figure 16c) handles the classification of above-ground elements, leading to the full semantic segmentation of Scene 1 (Figure 16d).



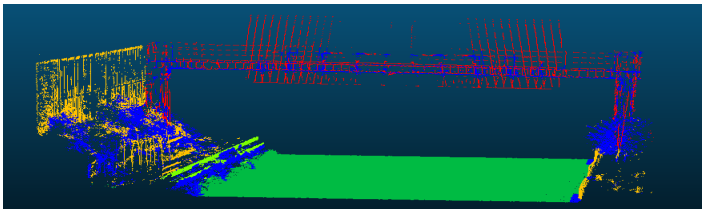
(**a**) Layer 1.



(**b**) Layer 2.



(**c**) Layer 3.



(**d**) Final scene

Figure 16. The semantic segmentation results of Straßen NRW dataset for Scene 1 using hierarchical SVM approach.

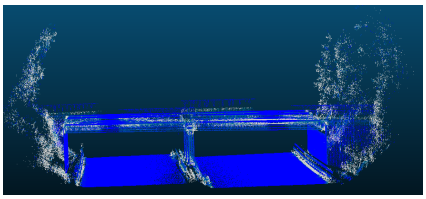
Table 5. Performance metrics of all classes for HSVM semantic segmentation approach for Scene 1.

Classes	Precision [%]	Recall [%]
vegetation	80.7	100
road	98.1	97.7
railing	97.2	67.6
noise barrier	95.4	94.63
signs	88.6	97.4

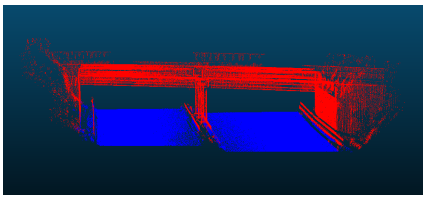
In the second scene, the number of classes is increased as a result of the inclusion of bridge elements, bringing the total to seven distinct classes. An in-depth analysis of the performance of the HSVM approach along with a detailed breakdown of evaluation metrics for each individual class are presented in Table 6. Notably, the overall accuracy achieved for this particular scene amounted to 90.27%. Figure 17 illustrates the HSVM model's performance on Scene 2 at different classification layers. In Figure 17a,b, the model distinguishes between vegetation and man-made objects as well as between ground and above-ground objects. However, in Layer 3 (Figure 17c), additional classes are classified compared to Scene 1 due to the presence of bridge elements. Figure 17d presents the final semantic segmentation for this dataset.

Table 6. Performance metrics of all classes for HSVM semantic segmentation approach for Scene 2.

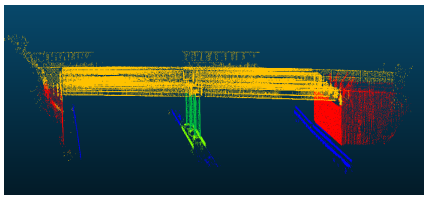
Classes	Precision [%]	Recall [%]
vegetation	90.00	81.67
road	95.52	100
railing	79.81	85.84
abutment	96.70	99.11
girder	79.61	100
pier	97.06	99.85
footing	94.01	90.93



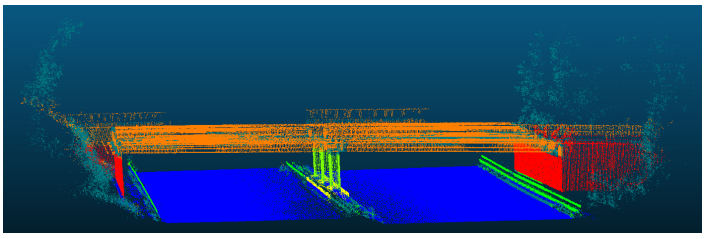
(**a**) Layer 1.



(**b**) Layer 2.



(**c**) Layer 3.



(**d**) Final scene.

Figure 17. The semantic segmentation results of Straßen NRW dataset for Scene 2 using hierarchical SVM approach.

6.2. Experiment 2: Deep Learning Methods

Given the inherent architecture of deep learning models, wherein features are autonomously extracted, the input feature vector for these models comprises only the raw x, y, and z coordinates of each point in the 3D point cloud. The definition of hyperparameters in these models significantly influences their performance.

6.2.1. Hyperparameters of PointNet and PointNet++ Models

- Learning rate: a hyperparameter that determines the speed at which the model learns from the training data and can have a significant impact on the accuracy and convergence of the model.
- Batch size: a hyperparameter that defines the number of training examples used in one iteration.

- Number of points in each batch: the number of points in each batch refers to the quantity of down-sampled points extracted from the original point cloud and presented to the model as batches.
- Epochs: an epoch refers to a single iteration through the entire training dataset during the training of a neural network.
- Optimizer: an optimizer adjusts the model's parameters during training to minimize a specified loss function, helping the network to learn and to improve its predictive accuracy; it does so by using gradients to update the parameters in a direction that reduces the loss.

Table 7 details the parameters employed for the training of the PointNet and PointNet++ models.

Table 7. Training parameters of PointNet and PointNet++ models.

Parameters	Values
learning rate	0.001
batch size	16
number of points in each batch	4096
epochs	200
optimizer	Adam

6.2.2. Hyperparameters of KPConv Model

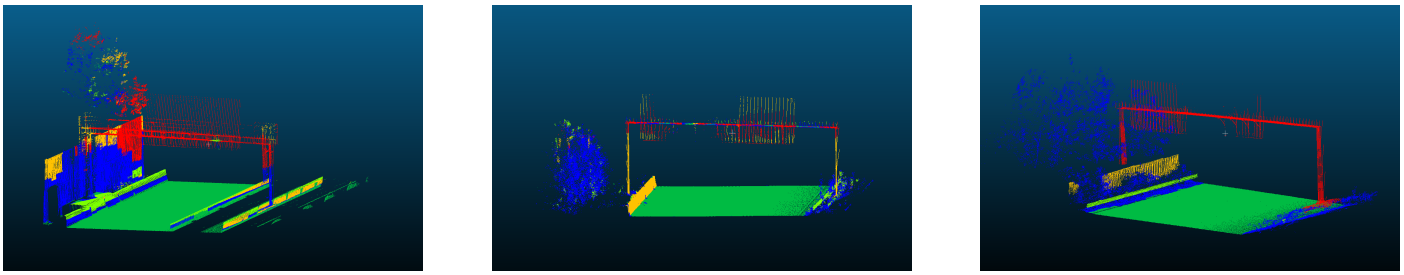
- Subsampling parameter (dl): a hyperparameter that controls the number of points to be subsampled at each layer in the network. The initial subsampling parameter is dl_0.
- Kernel points: a hyperparameter that defines the number of points within a local region to perform the convolution operation.
- Conv radius: a hyperparameter that determines the influence of each kernel point during the convolutional operation. It is crucial for the aggregation of information from kernel points to the points located within the defined radius.

Table 8 details the parameters employed for the training of the KPConv model.

Table 8. Training parameters of KPConv model.

Parameters	Values
learning rate	0.01
batch size	1
batch number	6
number of points in each batch	10,000
dl_0	0.06 m
epochs	200
kernel points	15 pts
conv radius	2.5 m
optimizer	momentum gradient descent

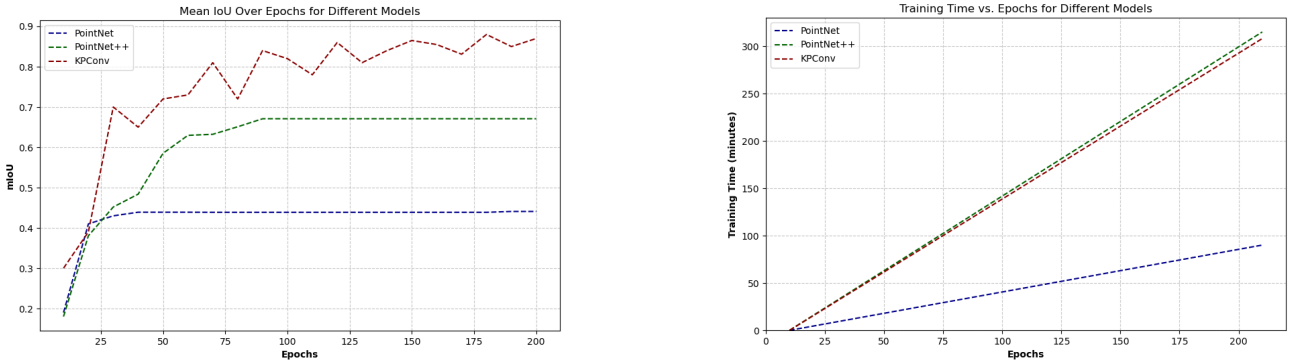
The semantic segmentation outcomes of the three deep learning models on the test dataset of Scene 1 (11,896,152 pts) are illustrated in Figures 18a, 18b, and 18c, respectively. Their evaluation metrics will be examined in comparison to our HSVM methodology.



(**a**) PointNet. (**b**) PointNet++. (**c**) KPConv.

Figure 18. Semantic segmentation results of the test datasets for Scene 1 with 5 classes using PointNet, PointNet++, and KPConv models.

To closely monitor the performance of the three models during training, Figure 19a displays the mean intersection over union (mIoU) for the validation datasets (2,186,724 points) across epochs, while Figure 19b illustrates the corresponding training times for each model. These detailed insights will prove invaluable in the forthcoming comparison between the HSVM and these models, as demonstrated in the following section.



(**a**) (**b**)

Figure 19. Performance analysis of the three investigated deep learning models during training process. (**a**) Monitoring the performance of the three models using the validation datasets across epochs. (**b**) Computational time required by each model during the training process.

7. Discussion

This section delves deeper into our developed methodology and compares it with the previously explored deep learning models.

7.1. HSVM Model

The methodology employing entropy as an energy function for optimizing neighborhood scale has proven effective for defining an optimal neighborhood and enables robust distinctions between various objects in infrastructure scenes. While planarity and sphericity features extracted from these neighborhoods effectively discern vegetation from man-made objects (Figure 13a,b), these approaches suffer from computational time constraints and dataset dependence. To address computational time, we developed an adaptive subsampling approach to reduce point cloud size and enhance computational efficiency. This adaptation facilitated the exploration of the geometric features of 1,000,000 pts across six scales (0.5 m to 3.5 m) in 65 s (Figure 13d). Features were then projected onto the original dataset of 2,187,299 pts in 45.1 s, resulting in a total processing time of 110.1 s. However, adaptive subsampling's effectiveness is highly dependent on the parameter $\Delta\rho$, which is selected through trial and error to balance computational time and segmentation accuracy. To tackle dataset dependence, a linear increase in the search radius balanced feature capture across small and large scales in infrastructure scenes.

Automated feature selection employing GA yielded highly favorable outcomes for delineating diverse feature sets in the first dataset. This proves valuable for identifying the features that significantly contribute to the model's performance, enabling a reduction in computational time. Notably, the results of the feature selection process revealed that the standard deviation of the height values was consistently excluded by the algorithm across all layers, as shown in Table 4. Conversely, the algorithm demonstrated early convergence for the third layer, specifically at Generation 2, as depicted in Figure 14. This highlights the robustness of the feature set for this layer via the exclusion of linearity features. This indicates that the features selected by the algorithm effectively differentiate above-ground elements, which include railings, noise barriers, and bridge signs and are characterized by both vertical and horizontal attributes.

Moreover, adopting a neighborhood labeling strategy based on the majority of points within it has significantly improved the hierarchical semantic segmentation approach (Figure 15). In many instances, the majority of points within a neighborhood inherently serve as the reference point; this is particularly noticeable for large objects observed from larger scales, like the pier class. However, for areas at the extremities of the pier—near other classes such as the bottom of bridge girders or adjacent to the footing—reference point labeling yields slightly better results, as depicted in Figure 15. Conversely, for classes observed at smaller scales, like vegetation and railings, labeling points based on the majority within the neighborhood yields better outcomes. This phenomenon is particularly evident for points situated on the boundaries of two classes, such as roads and roadside vegetation (Figure 15), where misclassification can occur if they are labeled solely according to a reference point.

The HSVM method proved reasonably effective at accurately segmenting and categorizing various objects within the scene and achieved average accuracies of 91.94% and 90.27% for the first and second datasets, respectively (refer to Figures 16 and 17). In Figure 16d, most vegetation is correctly classified despite being present in occluded areas behind noise barriers alongside the roadside. However, due to the sphericity of certain parts of the bridge signs, some points are misclassified as vegetation and are subsequently excluded after the first layer. Conversely, Figure 17d reveals that certain girders and the railing are erroneously classified as vegetation. This misclassification occurs primarily in densely vegetated areas where accurately extracting characteristic features of these elements proves challenging. Additionally, some girder points scatter in space at smaller scales, leading to misclassification as vegetation (Figure 17a).

The success of HSVM arises from optimizing each stage of the semantic segmentation workflow by utilizing an optimal neighborhood to extract diverse features and automatically selecting feature sets hierarchically. Hierarchical classification efficiently distinguishes between categories using various SVM techniques like one vs. one and one vs. rest, as shown in Figure 20. The model demonstrates high confidence in distinguishing ground and above-ground objects at Layer 2, with points at their interface showing a confidence level of approximately 0.5. Above-ground points exhibit a confidence level of approximately 1.0, while ground points show notably lower confidence. The one vs. one technique with a linear kernel enhances category distinction in this layer.

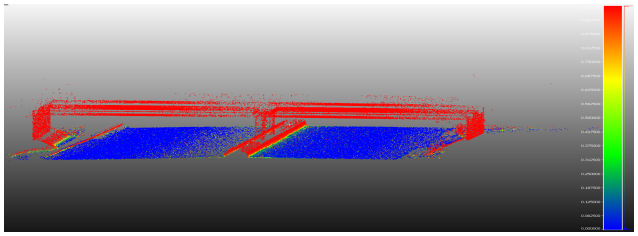


Figure 20. Probabilistic assessment of ground and above-ground object discrimination in Layer 2 for the HSVM model.

7.2. Comparison with Deep Learning Models

We compared the HSVM-based semantic segmentation approach with state-of-the-art deep learning models to determine its relative performance and positioning in the field. In this comparison, we took five key aspects into account to ensure a comprehensive and equitable evaluation. These are model accuracy, computational time, available computational resources, amount of data required for training, and the complexity of adjusting the hyperparameters for each model.

In terms of model accuracy, our HSVM-based semantic segmentation approach outperformed the PointNet and PointNet++ deep learning models, though it slightly lagged behind the more robust KPConv model. A comprehensive comparison of these four models on the test datasets for Scene 1, as depicted in Table 9 and Figure 21, highlights their effectiveness at predicting roads with high accuracy, mainly because of the substantial training data available for this class within the infrastructure scene. Notably, KPConv exhibited better performance across all classes in the scene, which is attributed to its capability to extract both low- and high-level features using convolutional operations.

Table 9. IoU per class for semantic segmentation on Dataset 1 across all models.

Method	Vegetation	Road	Railing	Noise Barrier	Signs	mIoU	OA
PointNet	30.1	93.5	31.7	11.8	53.3	44.1	82.5
PointNet++	54.4	96.2	57.2	65.5	42.3	56.7	89.8
HSVM	65.2	96.8	65.1	75.3	63.6	73.2	91.9
KPConv	69.4	97.1	74.2	84.5	89.3	82.9	95.9

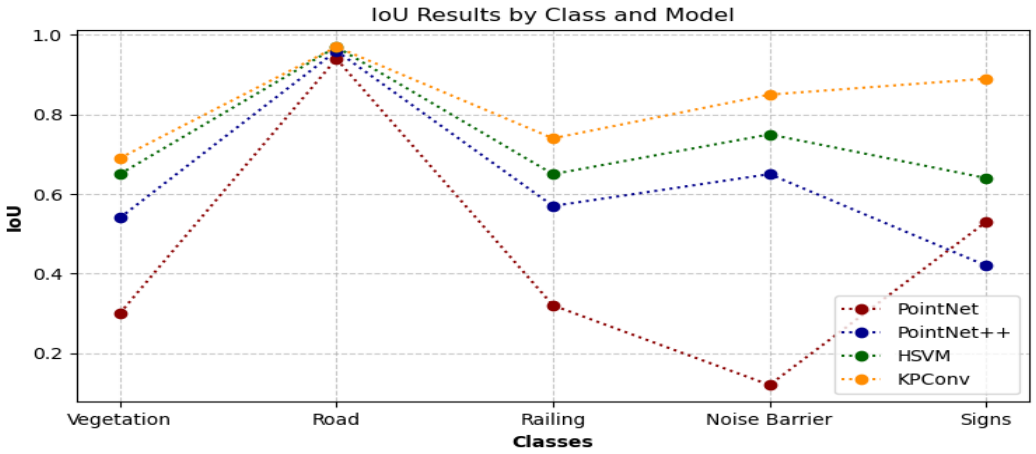


Figure 21. Performance of all investigated models on dataset for Scene 1 containing 5 classes.

In terms of computational time for feature selection and extraction, model training, and model testing, an overview is presented in Table 10, which outlines the time requirements for each applied method. It is worth noting that the time for feature processing is encompassed within the training duration of the HSVM and deep learning models. In the context of HSVM, the processing time required to identify the optimal neighborhood for 100,000 pts and 209,400 pts was 3.5 s and 15.5 s, respectively. Additionally, the time required by the genetic algorithm to define the optimal feature sets was 3 min. It is also noteworthy that the HSVM model utilized CPU resources more efficiently and with less computational time compared to other deep learning models, which typically require GPU resources and longer durations.

Table 10. Computational times for all investigated models.

Method	Training Time	Inference Time	Device
PointNet	≈1.5 h	≈0.32 h	GPU
PointNet++	≈5.25 h	≈0.37 h	GPU
HSVM	≈0.09 h	≈0.1 h	CPU
KPConv	≈5.13 h	≈1 h	GPU

A deeper analysis of the time data presented in Table 10 is necessary to take into account variables like data volume and the training progress of each deep learning model across epochs. Some models showed limited improvement with additional epochs, resulting in inefficient use of time. To address this issue, adjustments should be made to discount such times and compare durations regardless of dataset size. As a solution, we suggest introducing another comparison metric: **point prediction time (PPT)**. This metric assesses the total time required to process one thousand data points, offering insight into the efficiency of each model in handling specific data quantities. It proves particularly valuable in scenarios involving large datasets, where faster processing times are desirable.

$$\text{sec}/1000 \text{ pts} = 1000 \times \frac{\text{Training Time} + \text{ Inference Time}}{\text{Total number of points}} \tag{24}$$

Upon analyzing Figure 19a,b, it is apparent that the training evaluations of PointNet and PointNet++ reach a plateau after a certain number of epochs: 50 epochs for PointNet and 100 epochs for PointNet++. In contrast, KPConv consistently shows progress throughout the entire training process, peaking at the 180th epoch. Consequently, we only consider the training times corresponding to these epochs in the comparison, which are 8 min, 167 min, and 308 min, respectively. Using Equation (24), we estimate the point prediction time required by all the models under investigation. Table 11 illustrates the time needed by each model for feature extraction, model training, and the evaluation of 1000 pts. Deep learning models are known for their efficiency at processing batches of points during forward and backward propagation within the network. As the complexity of the network increases in terms of layers, parameters, etc., the computational time also rises, as evidenced by the lower values of PointNet, with its simpler architecture design compared to PointNet++ and KPConv, which have more complex designs.

Table 11. PPT required by each model to process a batch of 1000 pts.

Method	Training Data (pts)	Test Data (pts)	Time (sec/1000 pts)
PointNet	61,995,915	11,896,152	0.022
PointNet++	61,995,915	11,896,152	0.154
HSVM	100,000	209,400	2.21
KPConv	61,995,915	11,896,152	0.3

Optimizing the hyperparameters introduced in Section 6.2 is crucial to improve the performance of the deep learning models. A high learning rate may cause the model to diverge, while a low learning rate may result in slow convergence. The optimal batch size depends on the dataset size, memory, and model complexity. Increasing the number of epochs can improve accuracy on the training set, but overfitting may occur if the number of epochs is too high. Balancing these parameters is a challenging yet crucial task to achieve optimal performance of the deep learning models. This aspect should also be considered during the comparison between the investigated models.

Utilizing the predefined comparison criteria, each model undergoes qualitative evaluation within a ranking matrix employing a scoring system. This assessment employs "+" and "-" symbols tailored to specific factors. For instance, models with lower accuracy and higher computational time are attributed a lower positive score, while those with higher accuracy and lower time receive a higher positive score. Similarly, the amount of data used for training and computational requirements, including GPU or CPU usage, is qualitatively assigned either a "+" or "-" score. Larger datasets and intensive GPU usage qualitatively receive a "-" score. Notably, the adjustment of hyperparameters for deep learning models is inherently more intricate than for machine learning models, contributing to a complexity factor rated qualitatively with "-" symbols. Table 12 offers a thorough examination of the qualitative score matrix, outlining the strengths and weaknesses of each model. While the HSVM model excelled in four out of five criteria used for comparison, it falls short in terms of speed compared to deep learning models when handling a batch of 1000 points.

Table 12. Model ranking matrix based on various comparison aspects.

Method	PPT [1]	Computational Time	Accuracy	Data Volume	Device	Hyperparameters
PointNet	++++	+++	+	-	-	-
PointNet++	+++	++	++	-	-	-
HSVM [2]	**+**	**++++**	**+++**	**+**	**+**	**+**
KPConv	++	+	++++	-	-	-

[1] Point prediction time (sec/1000 pts). [2] Emphasizing the performance ranking of our approach relative to other deep learning models.

8. Conclusions

This research delves into automating the Scan2BIM workflow, with the primary aim of streamlining the labor-intensive process of 3D modeling, particularly for infrastructure scenes such as bridges. The Scan2BIM workflow comprises two main components: semantic segmentation and 3D modeling, both of which are potential candidates for automation. Our focus lies on automating the semantic segmentation of 3D point clouds captured by laser scanners by employing machine learning techniques: specifically, SVM models. We have developed our methodology with the goal of optimizing each step in the workflow to achieve the most accurate segmentation of 3D points.

The primary advancement in the developed workflow involves efficiently optimizing the neighborhood within a manageable computational timeframe; this acts as a preliminary segmentation phase to capture characteristic object features. We have assessed the limitations of this optimization process and proposed solutions to broaden its applicability to diverse datasets with unique traits. Subsequently, geometric and height features are extracted from the optimized neighborhood to construct a feature vector serving as training input for the model. However, we could not explore other feature types like texture features due to their unavailability across all datasets; their inclusion could enhance the classification model's robustness. Furthermore, we have integrated additional optimization techniques, such as genetic algorithms (GAs), during the feature selection stage to reduce feature dimensionality when training the SVM model. Lastly, hierarchical object classification within the scene has proven to be effective at categorizing the entire scene based on human-interpretable scene knowledge.

To evaluate the strengths and weaknesses of our HSVM approach, we conducted a comparative analysis against state-of-the-art deep learning models. Our findings reveal that HSVM exhibited high performance: outperforming renowned deep learning models like PointNet and PointNet++. Although HSVM did not quite match the performance of KPConv, it excelled in several areas, particularly in scenarios with limited training data. In such cases, HSVM can deliver highly satisfactory results within a short timeframe.

However, in terms of processing speed for point clouds, HSVM may not be as competitive as deep learning models.

This comparative analysis underscores the trade-off between speed and accuracy and aids researchers with choosing suitable approaches. It also paves the way for improved algorithms in semantic segmentation. Integrating automated features from deep learning like KPConv into models such as SVM could reduce computational time and enhance model interpretability—a field known as explainable machine learning—promising progress in point cloud segmentation.

The comparison highlights differences between machine learning and deep learning workflows, particularly for feature extraction. Machine learning uses handcrafted features for specific datasets, while deep learning automates this process, offering greater adaptability. This highlights trade-offs between manual intervention and automation and guides approach selection based on task requirements and resources.

Future studies will leverage semantically segmented point clouds for diverse 3D reconstruction techniques, automating the generation of geometric–semantic BIM models and thus streamlining the Scan2BIM workflow.

Author Contributions: Conceptualization, methodology, software, validation, formal analysis, investigation, data curation, writing—original draft preparation, visualization, writing—original draft: all by M.M.; writing—review and editing, M.M., J.M. and J.B; supervision, J.M. and J.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the German Research Foundation (DFG) as part of the Collaborative Research Center 339 (SFB/TRR 339) (project ID: 453596084) and the research project PointSemSeg+ (SPP 100+) (project ID: 501834640).

Data Availability Statement: The original contributions presented in the study are included in the article, further inquiries can be directed to the corresponding author .

Acknowledgments: The authors extend their sincere appreciation to Straßen NRW for generously providing access to the crucial datasets utilized in this study. The availability of these datasets has been instrumental to the successful execution of the methodologies described in the methods section.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

BIM	Building Information Modeling
SVM	Support Vector Machine
HSVM	Hierarchical Support Vector Machine
GA	Genetic Algorithm
PPT	Point Prediction Time

References

1. Bryde, D.; Broquetas, M.; Volm, J.M. The project benefits of Building Information Modelling (BIM). *Int. J. Proj. Manag.* **2013**, *31*, 971–980. [CrossRef]
2. Wong, A.; Wong, F.; Nadeem, A. Attributes of Building Information Modelling Implementations in Various Countries. *Archit. Eng. Des. Manag.* **2010**, *6*, 288–302. [CrossRef]
3. Heaton, J.; Parlikad, A.K.; Schooling, J. Design and development of BIM models to support operations and maintenance. *Comput. Ind.* **2019**, *111*, 172–186. [CrossRef]
4. Deng, M.; Menassa, C.C.; Kamat, V.R. From BIM to digital twins: A systematic review of the evolution of intelligent building representations in the AEC-FM industry. *J. Inf. Technol. Constr.* **2021**, *26*, 58–83. [CrossRef]
5. Vilgertshofer, S.; Mafipour, M.; Borrmann, A.; Martens, J.; Blut, T.; Becker, R.; Blankenbach, J.; Göbels, A.; Beetz, J.; Celik, F.; et al., TwinGen: Advanced technologies to automatically generate digital twins for operation and maintenance of existing bridges. In *ECPPM 2022-eWork and eBusiness in Architecture, Engineering and Construction 2022*; CRC Press: Boca Raton, FL, USA, 2023; pp. 213–220. [CrossRef]
6. Fukuoka, T.; Fujiu, M. Detection of Bridge Damages by Image Processing Using the Deep Learning Transformer Model. *Buildings* **2023**, *13*, 788. [CrossRef]

7. Adibfar, A.; Costin, A.M. Creation of a mock-up bridge digital twin by fusing intelligent transportation systems (ITS) Data into Bridge Information Model (BrIM). *J. Constr. Eng. Manag.* **2022**, *148*, 04022094. [CrossRef]
8. Iacovino, C.; Turksezer, Z.I.; Giordano, P.F.; Limongelli, M.P. Comparison of bridge inspection policies in terms of data quality. *J. Bridge Eng.* **2022**, *27*, 04021115. [CrossRef]
9. Nguyen, D.C.; Nguyen, T.Q.; Jin, R.; Jeon, C.H.; Shim, C.S. BIM-based mixed-reality application for bridge inspection and maintenance. *Constr. Innov.* **2022**, *22*, 487–503. [CrossRef]
10. Mohamed, A.G.; Khaled, A.; Abotaleb, I.S. A Bridge Information Modeling (BrIM) Framework for Inspection and Maintenance Intervention in Reinforced Concrete Bridges. *Buildings* **2023**, *13*, 2798. [CrossRef]
11. Ndekugri, I.; Braimah, N.; Gameson, R. Delay Analysis within Construction Contracting Organizations. *J. Constr. Eng. Manag.* **2008**, *134*, 692–700. [CrossRef]
12. Rodríguez-Gonzálvez, P.; Jiménez Fernández-Palacios, B.; Muñoz Nieto, A.L.; Arias-Sanchez, P.; Gonzalez-Aguilera, D. Mobile LiDAR System: New Possibilities for the Documentation and Dissemination of Large Cultural Heritage Sites. *Remote Sens.* **2017**, *9*, 189. [CrossRef]
13. Huang, Z.; Wen, Y.; Wang, Z.; Ren, J.; Jia, K. Surface Reconstruction from Point Clouds: A Survey and a Benchmark. *arXiv* **2022**, arXiv:2205.02413. [CrossRef]
14. Sharma, R.; Abrol, P. Parameter Extraction and Performance Analysis of 3D Surface Reconstruction Techniques. *Int. J. Adv. Comput. Sci. Appl.* **2023**, *14*, 331–336. [CrossRef]
15. Pătrăucean, V.; Armeni, I.; Nahangi, M.; Yeung, J.; Brilakis, I.; Haas, C. State of research in automatic as-built modelling. *Adv. Eng. Inform.* **2015**, *29*, 162–171. [CrossRef]
16. Iglesias, J.L.; Severiano, J.A.D.; Amorocho, P.E.L.; del Val, C.M.; Gómez-Jáuregui, V.; García, O.F.; Royano, A.P.; González, C.O. Revision of Automation Methods for Scan to BIM. In *Advances in Design Engineering: Proceedings of the XXIX International Congress INGEGRAF, LogroÃśo, Spain, 20–21 June 2019*; Springer International Publishing: Berlin/Heidelberg, Germany, 2019.
17. Ariyachandra, M.; Brilakis, I. Understanding the challenge of digitally twinning the geometry of existing rail infrastructure. In Proceedings of the 12th FARU International Research Conference, Colombo, Sri Lanka, 3–4 December 2019. [CrossRef]
18. Gourguechon, C.; Macher, H.; Landes, T. Automation of As-Built Bim Creation from Point Cloud: An Overview of Research Works Focused on Indoor Environment. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2022**, *XLIII-B2-2022*, 193–200. [CrossRef]
19. Zhang, G.; Vela, P.A.; Karasev, P.; Brilakis, I. A sparsity-inducing optimization-based algorithm for planar patches extraction from noisy point-cloud data. *Comput.-Aided Civ. Infrastruct. Eng.* **2015**, *30*, 85–102. [CrossRef]
20. Dimitrov, A.; Gu, R.; Golparvar-Fard, M. Non-uniform B-spline surface fitting from unordered 3D point clouds for as-built modeling. *Comput.-Aided Civ. Infrastruct. Eng.* **2016**, *31*, 483–498. [CrossRef]
21. Xu, Y.; Tuttas, S.; Hoegner, L.; Stilla, U. Voxel-based segmentation of 3D point clouds from construction sites using a probabilistic connectivity model. *Pattern Recognit. Lett.* **2018**, *102*, 67–74. [CrossRef]
22. Lu, R.; Brilakis, I.; Middleton, C.R. Detection of structural components in point clouds of existing RC bridges. *Comput.-Aided Civ. Infrastruct. Eng.* **2019**, *34*, 191–212. [CrossRef]
23. Zhang, G.; Vela, P.; Brilakis, I. Automatic generation of as-built geometric civil infrastructure models from point cloud data. In *Computing in Civil and Building Engineering*; American Society of Civil Engineers: Reston, VA, USA, 2014; pp. 406–413.
24. Ruizhongtai Qi, C.; Su, H.; Mo, K.; Guibas, L. PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, Las Vegas, NV, USA, 27–30 June 2016.
25. Thomas, H.; Qi, C.R.; Deschaud, J.E.; Marcotegui, B.; Goulette, F.; Guibas, L.J. KPConv: Flexible and Deformable Convolution for Point Clouds. *arXiv* **2019**, arXiv:1904.08889. [CrossRef]
26. Lu, R.; Brilakis, I. Digital twinning of existing reinforced concrete bridges from labelled point clusters. *Autom. Constr.* **2019**, *105*, 102837. [CrossRef]
27. Lu, R. Automated Generation of Geometric Digital Twins of Existing Reinforced Concrete Bridges. Ph.D. Thesis, University of Cambridge, Cambridge, UK, 2018. [CrossRef]
28. Mafipour, M.S.; Vilgertshofer, S.; Borrmann, A. Automated geometric digital twinning of bridges from segmented point clouds by parametric prototype models. *Autom. Constr.* **2023**, *156*, 105101. [CrossRef]
29. Martens, J.; Blankenbach, J. VOX2BIM+—A Fast and Robust Approach for Automated Indoor Point Cloud Segmentation and Building Model Generation. *PFG J. Photogramm. Remote Sens. Geoinf. Sci.* **2023**, 91, 273–294. [CrossRef]
30. Martens, J.; Blankenbach, J. An evaluation of pose-normalization algorithms for point clouds introducing a novel histogram-based approach. *Adv. Eng. Inform.* **2020**, *46*, 101132. [CrossRef]
31. Thomson, C.; Boehm, J. Automatic Geometry Generation from Point Clouds for BIM. *Remote Sens.* **2015**, *7*, 11753–11775. [CrossRef]
32. Grilli, E.; Menna, F.; Remondino, F. a Review of Point Clouds Segmentation and Classification Algorithms. *ISPRS Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2017**, *42W3*, 339–344. [CrossRef]
33. Li, Q.; Yuan, P.; Lin, Y.; Tong, Y.; Liu, X. Pointwise classification of mobile laser scanning point clouds of urban scenes using raw data. *J. Appl. Remote Sens.* **2021**, *15*, 024523. [CrossRef]
34. Lu, H.; Shi, H. Deep Learning for 3D Point Cloud Understanding: A Survey. *arXiv* **2020**, arXiv:2009.08920. [CrossRef]

35. Mukhamediev, R.I.; Symagulov, A.; Kuchin, Y.; Yakunin, K.; Yelis, M. From Classical Machine Learning to Deep Neural Networks: A Simplified Scientometric Review. *Appl. Sci.* **2021**, *11*, 5541. [CrossRef]
36. Thomas, H.; Goulette, F.; Deschaud, J.E.; Marcotegui, B.; LeGall, Y. Semantic Classification of 3D Point Clouds with Multiscale Spherical Neighborhoods. In Proceedings of the 2018 International Conference on 3D Vision (3DV), Verona, Italy, 5–8 September 2018; pp. 390–398. [CrossRef]
37. Weinmann, M.; Jutzi, B.; Hinz, S.; Mallet, C. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS J. Photogramm. Remote Sens.* **2015**, *105*, 286–304. [CrossRef]
38. Lee, I.; Schenk, T. Perceptual organization of 3D surface points. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2002**, *34*, 193–198.
39. Filin, S.; Pfeifer, N. Neighborhood Systems for Airborne Laser Data. *Photogramm. Eng. Remote Sens.* **2005**, *71*, 743–755. [CrossRef]
40. Linsen, L.; Prautzsch, H. Local Versus Global Triangulations. In Proceedings of the Eurographics 2001—Short Presentations, Eurographics Association, Manchester, UK, 3–7 September 2001. [CrossRef]
41. Niemeyer, J.; Rottensteiner, F.; Soergel, U. Contextual classification of lidar data and building object detection in urban areas. *ISPRS J. Photogramm. Remote Sens.* **2014**, *87*, 152–165. [CrossRef]
42. Brodu, N.; Lague, D. 3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology. *ISPRS J. Photogramm. Remote Sens.* **2012**, *68*, 121–134. [CrossRef]
43. Hackel, T.; Wegner, J.D.; Schindler, K. Fast Semantic Segmentation of 3d Point Clouds with Strongly Varying Density. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2016**, *3*, 177–184. [CrossRef]
44. Weinmann, M.; Jutzi, B.; Mallet, C. Semantic 3D scene interpretation: A framework combining optimal neighborhood size selection with relevant features. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2014**, *II3*, 181–188. [CrossRef]
45. Demantké, J.; Mallet, C.; David, N.; Vallet, B. Dimensionality Based Scale Selection in 3D Lidar Point Clouds. *ISPRS Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2012**, *XXXVIII-5/W12*, 97–102. [CrossRef]
46. Pauly, M.; Keiser, R.; Gross, M. Multi-scale Feature Extraction on Point-Sampled Surfaces. *Comput. Graph. Forum* **2003**, *22*, 281–289. [CrossRef]
47. Blomley, R.; Jutzi, B.; Weinmann, M. Classification of Airborne Laser Scanning Data Using Geometric Multi-Scale Features and Different Neighbourhood Types. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2016**, *III-3*, 169–176. [CrossRef]
48. Vosselman, G.; Gorte, B.G.; Sithole, G.; Rabbani, T. Recognising structure in laser scanner point clouds. *Inter. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2004**, *46*, 33–38.
49. Blomley, R.; Weinmann, M.; Leitloff, J.; Jutzi, B. Shape distribution features for point cloud analysis and A geometric histogram approach on multiple scales. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2014**, *II-3*, 9–16. [CrossRef]
50. Jutzi, B.; Groß, H. Nearest Neighbour Classification on Laser Point Clouds to Gain Object Structures from Buildings. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2009**, *38*, 4–7.
51. Weinmann, M.; Jutzi, B.; Mallet, C. Feature relevance assessment for the semantic interpretation of 3D point cloud data. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2013**, *II-5/W2*, 313–318. [CrossRef]
52. Chehata, N.; Guo, L.; Mallet, C. Airborne Lidar Feature Selection for Urban Classification Using Random Forests. In Proceedings of the Laserscanning, Paris, France, 1–2 September 2009.
53. Guyon, I.; Elisseeff, A. An introduction to variable and feature selection. *J. Mach. Learn. Res.* **2003**, *3*, 1157–1182.
54. Chandrashekar, G.; Sahin, F. A survey on feature selection methods. *Comput. Electr. Eng.* **2014**, *40*, 16–28. [CrossRef]
55. Khanum, M.; Mahboob, T.; Imtiaz, W.; Ghafoor, H.A.; Sehar, R. A survey on unsupervised machine learning algorithms for automation, classification and maintenance. *Int. J. Comput. Appl.* **2015**, *119*, 34–39. [CrossRef]
56. Chen, L.; Zhang, Y.; Lin, Y.; Jiang, M.; Huang, Y.; Lei, Y. Consistency-Based Semi-Supervised Learning for Point Cloud Classification. In Proceedings of the 2021 4th International Conference on Pattern Recognition and Artificial Intelligence (PRAI), Yibin, China, 20–22 August 2021; pp. 440–445. [CrossRef]
57. Van Engelen, J.E.; Hoos, H.H. A survey on semi-supervised learning. *Mach. Learn.* **2020**, *109*, 373–440. [CrossRef]
58. Park, Y.; Guldmann, J.M. Creating 3D City Models with Building Footprints and LiDAR Point Cloud Classification: A Machine Learning Approach. *Comput. Environ. Urban Syst.* **2019**, *75*, 76–89. [CrossRef]
59. Vapnik, V. *The Nature of Statistical Learning Theory*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
60. Hu, J.; Li, D.; Duan, Q.; Yueqi, H.; Chen, G.; Si, X. Fish species classification by color, texture and multi-class support vector machine using computer vision. *Comput. Electron. Agric.* **2012**, *88*, 133–140. [CrossRef]
61. Zhang, Y.D.; Wu, L. Classification of Fruits Using Computer Vision and a Multiclass Support Vector Machine. *Sensors* **2012**, *12*, 12489–505. [CrossRef]
62. Osisanwo, F.; Akinsola, J.; Awodele, O.; Hinmikaiye, J.; Olakanmi, O.; Akinjobi, J. Supervised machine learning algorithms: Classification and comparison. *Int. J. Comput. Trends Technol. (IJCTT)* **2017**, *48*, 128–138.
63. Murty, M.N.; Raghava, R. Kernel-Based SVM. In *Support Vector Machines and Perceptrons: Learning, Optimization, Classification, and Application to Social Networks*; Springer International Publishing: Cham, Switerland, 2016; pp. 57–67. [CrossRef]
64. Cervantes, J.; Garcia-Lamont, F.; Rodríguez-Mazahua, L.; Lopez, A. A comprehensive survey on support vector machine classification: Applications, challenges and trends. *Neurocomputing* **2020**, *408*, 189–215. [CrossRef]

65. Hachimi, M.; Kaddoum, G.; Gagnon, G.; Illy, P. Multi-stage Jamming Attacks Detection using Deep Learning Combined with Kernelized Support Vector Machine in 5G Cloud Radio Access Networks. In Proceedings of the 2020 International Symposium on Networks, Computers and Communications, Montreal, QC, Canada, 20–22 October 2020.
66. Rivolli, A.; Read, J.; Soares, C.; Pfahringer, B.; de Carvalho, A.C. An empirical analysis of binary transformation strategies and base algorithms for multi-label learning. *Mach. Learn.* **2020**, *109*, 1509–1563. [CrossRef]
67. Chen, C.; Li, X.; Belkacem, A.N.; Qiao, Z.; Dong, E.; Tan, W.; Shin, D. The Mixed Kernel Function SVM-Based Point Cloud Classification. *Int. J. Precis. Eng. Manuf.* **2019**, *20*, 737–747. [CrossRef]
68. Anandakumar, R.; Nidamanuri, R.; Krishnan, R. Semantic labelling of Urban point cloud data. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2014**, *40*, 907–911. [CrossRef]
69. Madzarov, G.; Gjorgjevikj, D. Multi-class classification using support vector machines in decision tree architecture. In Proceedings of the IEEE EUROCON 2009, St. Petersburg, Russia, 18–23 May 2009; pp. 288–295. [CrossRef]
70. He, Y.; Yu, H.; Liu, X.; Yang, Z.; Sun, W.; Wang, Y.; Fu, Q.; Zou, Y.; Mian, A. Deep learning based 3D segmentation: A survey. *arXiv* **2021**, arXiv:2103.05423.
71. Guo, Y.; Wang, H.; Hu, Q.; Liu, H.; Liu, L.; Bennamoun, M. Deep learning for 3d point clouds: A survey. *IEEE Trans. Pattern Anal. Mach. Intell.* **2020**, *43*, 4338–4364. [CrossRef] [PubMed]
72. Cai, Y.; Huang, H.; Wang, K.; Zhang, C.; Fan, L.; Guo, F. Selecting Optimal Combination of Data Channels for Semantic Segmentation in City Information Modelling (CIM). *Remote Sens.* **2021**, *13*, 1367. [CrossRef]
73. Hu, Q.; Yang, B.; Xie, L.; Rosa, S.; Guo, Y.; Wang, Z.; Trigoni, N.; Markham, A. RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds. *arXiv* **2020**, arXiv:1911.11236. [CrossRef]
74. Zhang, Z.; Hua, B.S.; Yeung, S.K. ShellNet: Efficient Point Cloud Convolutional Neural Networks using Concentric Shells Statistics. *arXiv* **2019**, arXiv:1908.06295. [CrossRef]
75. Landrieu, L.; Simonovsky, M. Large-scale Point Cloud Semantic Segmentation with Superpoint Graphs. *arXiv* **2018**, arXiv:1711.09869. [CrossRef]
76. Qu, T.; Di, S.; Feng, Y.T.; Wang, M.; Zhao, T.; Wang, M. Deep Learning Predicts Stress-Strain Relations of Granular Materials Based on Triaxial Testing Data. *Comput. Model. Eng. Sci.* **2021**, *128*, 129–144. [CrossRef]
77. Hinks, T.; Carr, H.; Truong-Hong, L.; Laefer, D.F. Point cloud data conversion into solid models via point-based voxelization. *J. Surv. Eng.* **2013**, *139*, 72–83. [CrossRef]
78. Yao, X.; Guo, J.; Hu, J.; Cao, Q. Using Deep Learning in Semantic Classification for Point Cloud Data. *IEEE Access* **2019**, *7*, 37121–37130. [CrossRef]
79. Qi, C.; Yi, L.; Su, H.; Guibas, L.J. PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Adv. Neural Inf. Process. Syst.* **2017**, *30*, 5105–5114.
80. Thompson, N.C.; Greenewald, K.; Lee, K.; Manso, G.F. The Computational Limits of Deep Learning. *arXiv* **2022**, arXiv:2007.05558. [CrossRef]
81. Zhang, M.; You, H.; Kadam, P.; Liu, S.; Kuo, C.C.J. PointHop: An Explainable Machine Learning Method for Point Cloud Classification. *IEEE Trans. Multimed.* **2020**, *22*, 1744–1755. [CrossRef]
82. Dogan, Ü.; Edelbrunner, J.; Iossifidis, I. Autonomous driving: A comparison of machine learning techniques by means of the prediction of lane change behavior. In Proceedings of the 2011 IEEE International Conference on Robotics and Biomimetics, Beach, Thailand, 7–11 December 2011; pp. 1837–1843. [CrossRef]
83. Saraiva, P. On Shannon entropy and its applications. *Kuwait J. Sci.* **2023**, *50*, 194–199. [CrossRef]
84. Shannon, C.E. A mathematical theory of communication. *Bell Syst. Tech. J.* **1948**, *27*, 379–423. [CrossRef]
85. Gaikwad, A.; Datta, D.; Sharma, P.; Bera, S. Application of Shannon Entropy to Optimize the Data Analysis. In Proceedings of the International Conference on Advanced Engineering Optimization through Intelligent Techniques, Surat, India, 1–3 July 2013.
86. Goldberg, D.E. *Genetic Algorithms*; Pearson Education India: Noida, India, 2013.
87. Brindle, A. Genetic Algorithms for Function Optimisation. Ph.D. Thesis, Department of Computing Science, University of Alberta, Edmonton, AB, Canada, 1981.
88. Pedregosa, F.; Varoquaux, G.; Gramfort, A.; Michel, V.; Thirion, B.; Grisel, O.; Blondel, M.; Prettenhofer, P.; Weiss, R.; Dubourg, V.; et al. Scikit-learn: Machine Learning in Python. *J. Mach. Learn. Res.* **2011**, *12*, 2825–2830.
89. Hunter, J.D. Matplotlib: A 2D graphics environment. *Comput. Sci. Eng.* **2007**, *9*, 90–95. [CrossRef]
90. Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*; Curran Associates, Inc.: Glasgow, UK, 2019; pp. 8024–8035.