3D Point Cloud Semantic Segmentation Through Functional Data Analysis

Manuel OVIEDO DE LA FUENTE🟢, Carlos CABO, Javier ROCA- PARDIÑAS, E. Louise LOUDERMILK, and Celestino ORDÓÑEZ

Here, we propose a method for the semantic segmentation of 3D point clouds based on functional data analysis. For each point of a training set, a number of handcrafted features representing the local geometry around it are calculated at different scales, that is, varying the spatial extension of the local analysis. Calculating the scales at small intervals allows each feature to be accurately approximated using a smooth function and, for the problem of semantic segmentation, to be tackled using functional data analysis. We also present a step-wise method to select the optimal features to include in the model based on the calculation of the distance correlation between each feature and the response variable. The algorithm showed promising results when applied to simulated data. When applied to the semantic segmentation of a point cloud of a forested plot, the results proved better than when using a standard multiscale semantic segmentation method. The comparison with two popular deep learning models showed that our proposal requires smaller training samples sizes and that it can compete with these methods in terms of prediction.

Key Words: Laser scanning; Multiscale analysis; Functional data; Multiclass classification; Variable selection.

1. INTRODUCTION

Semantic segmentation of 3D point clouds using machine learning involves assigning a label to each point in a point cloud a set of features obtained from the coordinates of a training sample (sometimes color and/or intensity are also used as features) taken as explanatory

M. Oviedo de la Fuente (✉) Department of Mathematics, CITIC, Facultad de Informática, University of A Coruña, Campus de Elviña s/n, 15071 A Coruña, Spain. (E-mail: *manuel.oviedo@udc.es*).

C. Cabo, Faculty of Science and Engineering, Swansea University, Singleton Campus, Swansea, UK.

C. Cabo, Department of Mining Exploitation and Prospecting, University of Oviedo, Escuela Politécnica de Mieres, 33600 Mieres, Spain.

C. Ordóñez, Department of Mining Exploitation and Prospecting, University of Oviedo, Escuela Politécnica de Mieres, 33600 Mieres, Spain.

J. Roca-Pardiñas, Department of Statistics, The Galician Centre for Mathematical Research and Technology (CIT-MAga), University of Vigo, Vigo, Spain.

E. L. Roca-Pardiñas, Center for Forest Disturbance Science, USDA Forest Service Southern Research Station, 320 Green Street, 30602 Athens, GA, USA.

Published online: 12 September 2023

variables. These samples are often obtained through the analysis of the local geometry around each point on the training dataset. The results tend to improve when the features are calculated at different scales of local vicinity. This is usually accomplished using the points within spheres of different radii centered on each training point (Lee and Schenk 2002), or by taking data from a variable number of their nearest neighbors (Linsen and Prautzsch 2001) to calculate the covariance matrix and extract the eigenvalues and eigenvectors. Hence, the name multiscale machine learning (ML) semantic segmentation has been the object of research for many years and has largely replaced rule-based methods (Guo et al. 2015; Cabo et al. 2019; Xie et al. 2020). Nowadays, its use is so widespread that it is considered a standard method for 3D point cloud semantic segmentation. In fact, such methods have been implemented in a range of widely used software for 3D point cloud visualization and analysis, such as the CANUPO suite implemented in Cloud Compare (Brodu and Lague 2012). In recent years, the interest has moved towards deep learning (DL) (Zhang et al. 2019). PointNet (Qi et al. 2017) is a seminal DL model for point cloud classification and semantic segmentation that does not use approaches such as voxelization or projections onto planes. PointNet was improved using a hierarchical approach called PointNet++ (Qi et al. 2017). Since then, new DL models for semantic segmentation have emerged, such as DGCNN (Wang et al. 2018), SEGCloud (Tchapmi et al. 2017), RandLA-Net (Hu et al. 2021), cylindrical and asymmetrical 3D convolution networks (Xinge et al. 2021) or Point Transformer (Zhao et al. 2021), among others. Also, a set of benchmark datasets have been built to compare those models such as Shapenet (Chang et al. 2015), Scannet (Dai et al. 2017) or Semantickitti (Behley et al. 2019). DL may be favorable because handcrafted features are not needed (although results can be improved by using them) and that it generally outperforms ML methods in terms of providing more accurate results for large training samples. However, DL also has some drawbacks: (1) in general, it requires much more training data, so considerably more time to label the point clouds, (2) it uses models with several hidden layers and many parameters that are difficult to design and which are prone to overfitting, (3) the models are like black boxes, hence very difficult to interpret, and (4) hardware requirements are also greater and GPU processing is common in order to reduce computing time. For these reasons, semantic segmentation with ML is still competitive and, in some aspects, such as interpretability, better than DL, so further research in this area is worth pursuing.

One of the issues concerning the use of ML for semantic segmentation is finding the scales at which optimal results are obtained. The issue is not straightforward, and several solutions have been proposed, none of them definitive. One simple and common option, but inaccurate, is to select scales at regular intervals, that is, according to a linear function. A slightly better method is to use a quadratic function, since short scales are often more informative than large scales (Demantké et al. 2011). Obviously, neither method mentioned ensures that the most suitable scales are selected. A more sophisticated approach consists of taking into account the point density and the curvature at each point, as it was analyzed previously for noisy point clouds (Mitra and Nguyen 2023). A different solution (Weinmann et al. 2015) is based on analyzing the structure of the local covariance matrix obtained from the coordinates of the points and a measure of the uncertainty, such as Shannon's entropy (Shannon 1948). Recently, Oviedo-de la Fuente et al. (2021) has proposed estimating the optimum scales

being the maximum values of the distance correlation (DC) functions between the features and the label assigned to each point. Comparing this approach with other methods using a real dataset revealed the advantages of this approach, interpretability and predictive capacity. In this study, we decided to deal with the problem in a different way: instead of trying to find the optimal scales, we calculated the features at a large number of scales at equal intervals and considered them as scale-dependent functions. The segmentation is then performed as if infinite scales were used to calculate the features.

Another aspect of interest in ML is to ascertain the importance of the features, which has largely been studied using regression and classification. A review of different feature selection methods can be found in Jović et al. (2015). The goal is often to simplify the process by selecting only features relevant to the specific situation, which results in reduced computing time and more accurate results, as well as a better understanding of the relationship between the features and response variable. Some of the methods, such as that proposed in this work, select features according to their performance in a modelling algorithm (wrapper methods). Here, we present a fast and simple stepwise method based on a previous work for feature selection in regression (Febrero-Bande and González-Manteiga 2019) that progressively adds features to a model depending on the correlation between those features and the error in the model. This algorithm has been modified to be applied to a classification problem such as the one at hand. From now on, we will consider semantic segmentation as a synonym for classification in statistics although in point cloud processing they refer reference to different tasks).

The main novelty of this study lies in treating features calculated at different scales as functions instead of vectors, avoiding overlooking reporting scales as well as having to select the optimal scales, which can be a complex and time-consuming procedure.

The paper is structured as follows: Sect. 2 outlines our approach, which includes a method to select the most important features for the semantic segmentation of point clouds. Section 3 presents simulation studies aimed at evaluating the performance of the proposed methodology under different scenarios. Section 4 presents a case study using semantic segmentation to dissect a forested area into individual components and Sect. 5 focuses on our conclusions.

2. METHODOLOGY

2.1. STATEMENT OF THE PROBLEM

Let us consider a data sample of n observations $\{\mathcal{X}_i, Y_i\}_{i=1}^n$, where $\mathcal{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_p)$ is a vector of predictors and $Y \in \{1, 2, \ldots, C\}$, the variable that codifies the category assigned to each observation. Each predictor $\mathbf{X}_j \in \mathbb{R}^K$ is observed in a set of K discretization points, so $\mathbf{X}_j = \left(x_{j k_1}, \ldots, x_{j k_K} \right)$. In point cloud semantic segmentation, $\mathbf{X}_j$ represents each of the features used to segment the point cloud, $(k_1, k_2, \ldots, k_K)$ the scales of calculation of these features, and Y the label assigned to each of the points (Thomas et al. 2018; Atik and Duran 2021). As mentioned earlier, in this case the features are local geometric attributes, such as Linearity or Planarity, that change with the scale used, that is, with the size of the neighborhood around each point considered in their calculation. The objective of semantic segmentation is to assign a new observation to a specific class based on the particular

characteristics of that class obtained from the training data, thereby minimizing error in the assignment. This is achieved by means of a classifier that searches for patterns in the values of the features corresponding to each class.

Traditionally, features are vectors that contain information at multiple scales, but in this work we instead approximate them by using smooth functions. Thus, it is possible to analyze the features in a continuous range of scales and, in addition, to take advantage of the information provided by the derivatives of those functions. Theoretically, approximating features by smooth functions is a reasonable hypothesis for dense and uniform point clouds since small increments of the scale lead to small increments of the radius of the spheres centered in each point, that in turn will slightly modify the value of the features. This is favored by the fact that the spheres are concentric, so the larger spheres contain all the points of the smaller concentric spheres. However, in real situations the scale intervals are discrete and discontinuities in the features can be expected at small scales for irregular point clouds with sparse points in some areas containing different types of objects. In any case, even in this situation representing the features as functions is useful to reach a better understanding of the problem and also to discover discontinuities in the features.

Following a functional perspective, we will consider each calculated vector of features $\mathbf{X}_j, j = 1, \ldots, p$, as a sample with error of an underlying theoretical smooth function $X_j(k)$ on the real separable Hilbert space $H \equiv \mathcal{L}^2(\mathcal{K})$, defined on a compact interval $\mathcal{K}$ and endowed with inner product $\langle x, y \rangle = \int_{\mathcal{K}} x(k)y(k)\mathrm{d}k$ and norm $\|x\| = \int_{\mathcal{K}} x^2(k))\mathrm{d}k$.

Besides performing the semantic segmentation from a functional point of view, we also propose a methodology to select the most relevant features, i.e., those having a significant influence on the results of the semantic segmentation. Variable selection in multivariate analysis, including regression and classification, is a widely studied topic (Blum and Langley 1997; Kuhn and Johnson 2019), as it allows to simplify the models and offers a better comprehension of the solutions. In functional data classification, variable selection makes reference to replacing the function $X_j(k)$ with a lower dimensional vector (Fraiman et al. 2016; Berrendero et al. 2016); however, here we attempt to find a subset $X_l(k), l = 1, ..., q, q < p$, of the original features that results in a classification error close to (or even lower than) the error corresponding to a model that incorporates all the features.

There are many classification methods for multivariate or functional data, so it is not feasible to test them all. Besides, this is not being the purpose of this work. As such, in this study we have applied four well-known and tested methods, both multivariate and functional approaches: generalized linear model (GLM) (Hastie and Tibshirani 1987), random forest (RF) (Breiman 2001; Möller et al. 2016), support vector machines (SVM) (Boser et al. 1992; Rossi and Villa 2006) and a generalized linear model (GLM) with regularization (Friedman et al. 2010).

2.2. Generating functional features from point clouds

Feature engineering from point clouds is summarized in Fig. 1. For each point in the point cloud, a sphere of a specific radius (scale) centered in this point is created and the points inside this sphere are used to obtain a value of the feature at that scale. Then, the values of the features depend on the points and on the scale. As will be explained in Section 4,

Neighborhood selection

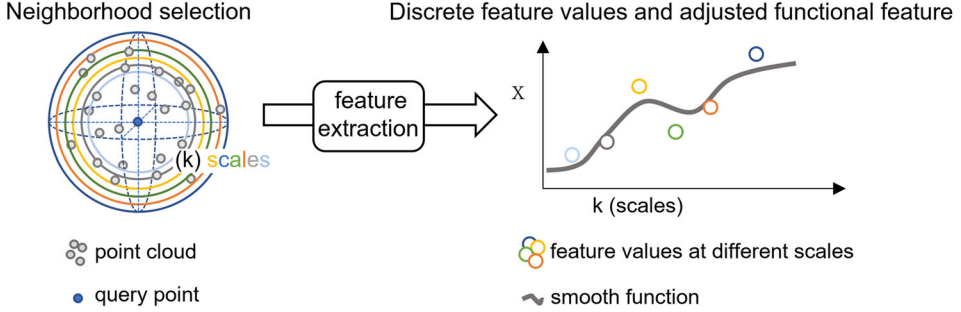Discrete feature values and adjusted functional feature

Figure 1. Scheme of the procedure to generate functional features from discrete features at different scales.

there are some typical features used in semantic segmentation obtained through a principal component analysis where the variables are the coordinates of the points inside the sphere. In a multiscale analysis, the procedure is repeated for concentric spheres of different radius centered on each point of the point cloud. Consequently, the larger spheres contain points of the smaller ones. As mentioned before, this implies that, except in exceptional cases (sparse points in border areas), there will be no abrupt changes in the values of the features as the radius of the spheres increases. Sampling a point cloud at different scales reveals different properties of the underlying surface, improving the semantic segmentation (Hackel et al. 2016).

Given the values of any feature at each point corresponding at different scales, a smooth function is adjusted as explained in the next section. As a result, each feature associated at the center of the sphere is now a function and, therefore, a functional data classification analysis can be conducted, the functions being the explanatory variables (features) and the label assigned to each of them the response variable.

2.3. Smoothing by decomposition in basis functions

A standard method for adjusting a smooth curve to the observations in functional data analysis is to consider functions as linear combinations of a finite number of basis functions. Basis functions can be of different types: Fourier basis, polynomials, B-splines (BSP) or wavelets, depending on the characteristics of the data. Nevertheless, the standard Karhunen-Loève decomposition is represented as follows:

$$\hat{X}_j(k) = \sum_{l=1}^{L} c_{jl} \phi_l(k) \tag{1}$$

where $c_{jl} \in \mathbb{R}$ are coefficients and $\phi_l(k) \in \mathcal{L}^2(\mathcal{K})$ the basis functions. The lower the number of basis L, the greater the smoothing and the greater the dimension reduction.

A well-known and widely used basis expansion is the Fourier series expansion:

$$\hat{X}(k) = c_0 + c_1 \sin wk + c_2 \cos wk + c_3 \sin 2wk + c_4 \cos 2wt + \dots$$

The basis functions are periodic sine and cosine functions, and the parameter w determines the period $2\pi/w$. When the values of k are equally spaced, then the basis is orthogonal. For

this and other aspects on Functional Data Analysis, it is advisable to consult (Ramsay and Silverman 1997).

The coefficients in (1) are usually determined by a least squares approach:

$$\sum_{m=1}^{K} \left[x_{j_{k_m}} - \sum_{l=1}^{L} c_{jl} \phi_l(k_m) \right]^2 = \left(\mathbf{X}_j - \Phi \mathbf{c}_j \right)^\top \left(\mathbf{X}_j - \Phi \mathbf{c}_j \right) = \| \mathbf{X}_j - \Phi \mathbf{c}_j \|^2 \quad (2)$$

where $\mathbf{X}_j^\top = \left(x_{j_{k_1}}, ... x_{j_{k_K}} \right)$, $\mathbf{c}_j^\top = \left(c_{j1}, ..., c_{jL} \right)$ and $\Phi = \{ \phi_1(k_m), ..., \phi_L(k_m) \}_{m=1}^{K}$ a $K \times L$ matrix. The solution to the minimization problem in (2) is:

$$\mathbf{c}_j = \left(\Phi^\top \Phi \right)^{-1} \Phi^\top \mathbf{X}_j \quad (3)$$

when Φ is $K \times L$ full rank matrix.

Calculating the inverse of the matrix in (3) can be computationally expensive for high dimension problems. In this case, it is advisable to generate band matrices or better still, diagonal matrices, as when orthogonal Fourier basis functions are used, given that $\Phi^\mathrm{T} \Phi$ is a diagonal matrix.

An alternative for smoothing wiggly curves is to add a penalization term to (3) , so the expression to be minimized is:

$$\sum_{m=1}^{K} \left[x_{j_{k_m}} - \sum_{j=1}^{L} c_{jl} \phi_l(k_m) \right]^2 + \lambda \cdot PEN(X_j) \quad (4)$$

where $\lambda \in \mathbb{R}$ is a smoothing parameter to fix the intensity of the penalty term $PEN(X_j)$. A popular penalty term is $PEN_2(X_j) = \int \left[\sum_{l=1}^{L} c_{jl} D^2 \left(\phi_l(k) \right) \right]^2 \mathrm{d}k$, which penalizes the curvature of the functions $X_j(k)$ through the calculation of the second derivative, D^2, of the basis functions. When λ is zero, the minimization problem reduces to minimize the square of the residuals, but as its values increase the penalty term becomes more important and the adjusted function tends to be smoother, with small second derivatives.

2.4. Decomposition in functional principal components

Similar to its multivariate counterpart, functional principal component analysis (FPCA) aims to obtain a small number of orthogonal functions that most efficiently describe the variations in the data Principal Component Analysis, functional or not, aims to find a lower-dimensional representation of the problem while preserving the maximum amount of information from the original variables. PCA results from the solution of the following eigenequation:

$$\hat{\Sigma}(\mathbf{X}) \hat{\xi} = \hat{\lambda} \hat{\xi} \quad (5)$$

where $\hat{\Sigma}$ is the covariance matrix of the data represented by the $n \times p$ dimensional matrix $\mathbf{X}$, while $\hat{\lambda} \in \mathbb{R}$ and $\hat{\xi} \in \mathbb{R}^p$ represent eigenvalues and eigenvectors, respectively.

The extension of PCA to FPCA consists in replacing vector by functions, matrices by linear operators and scalar products in a vector space by scalar functions in a square-integrable functional space (Han 2014). Accordingly, FPCA results from the solution of the Fredholm functional eigenequation (note the similarity with equation (5)):

$$\int_{\mathcal{K}} \hat{\Sigma}(s, k) \hat{\xi}(k) \, dk = \hat{\lambda} \hat{\xi}(s) \tag{6}$$

Assuming that the predictors $X_j, j = 1, \ldots, p$, are centered around the mean, the covariance function is estimated by (from now on, we will dispense with the subscripts to simplify):

$$\hat{\Sigma}(s, k) = n^{-1} \phi(s)^\top \mathbf{C}^\top \mathbf{C} \phi(k) \tag{7}$$

where $\mathbf{C}$ is an $n \times L$ matrix that stores the coefficients, $\phi(.)$ a column vector of length L, $\hat{\lambda} \in \mathbb{R}$ an eigenvalue, and $\hat{\xi}$ the corresponding orthogonal eigenfunction, verifying $\int_K \hat{\xi}_r(k)\hat{\xi}_s(k)dk = \delta_{r,s}$ for all r, s.

Similar to $X(k)$, each eigenfunction $\xi(s)$ has an expansion in basis functions

$$\hat{\xi}(s) = \sum_{l=1}^{L} b_l \phi_l(s) = \phi(s)^\top \mathbf{b} \tag{8}$$

where $\phi(s)^\top = (\phi_1, \ldots, \phi_L)$ and $\mathbf{b}^\top = (b_1, \ldots, b_L)$.

Substituting (6) and (7) in (5) results in the following matrix eigenequation:

$$n^{-1} \mathbf{C}^\top \mathbf{C} \mathbf{W} \mathbf{b} = \hat{\lambda} \mathbf{b} \tag{9}$$

where $\mathbf{W} = \int \phi^\top \phi$ is an $L \times L$ symmetric matrix of the inner products $\langle \phi_{l_1}, \phi_{l_2} \rangle = \int_K \phi_{l_1}(k)\phi_{l_2}(k)dk$.

The solution, $\mathbf{b}$, to this eigenequation contains the eigenvectors associated with the eigenvalues $\hat{\lambda}$ of the matrix $\mathbf{CW}$. When basis functions are orthonormal, $\mathbf{W} = \mathbf{I}$, FPCA is equivalent to a standard multivariate PCA applied to the matrix of coefficients $\mathbf{C}$ (Florence 2016).

2.5. Feature selection

One of the purposes of this research is to select, among the various features constructed from the coordinates of the points, those that make a significant contribution in terms of the results of the classification. The idea behind the method, which is independent of the classifier, is that the residuals of a model containing some of the features can be related to other features not included in the model, and that among these features the one most correlated with the residuals is the best candidate to add to the model in order to improve the results. If the model improves in terms of a metric for the classification, CM, (specifically in this work intersection-over-union, IoU, also named the Jaccard index, has been used as the metric), then feature is definitively incorporated into the model, if not, it is rejected. The procedure starts with a model that uses a single feature (the one with the highest distance

correlation with the vector of categories), and the rest of the features are progressively incorporated following the same criteria, their correlation with the residuals of the previous iteration and the accuracy or other metric used to evaluate the classification, until none of the features has a significant correlation with the output.

The proposed algorithm for variable selection is shown below (Algorithm 1). The main idea behind this iterative procedure is that the residuals of the classification can capture information not collected in previous steps. Firstly, three parameters are initialized: a set $M^{(i)}$ containing the features of the model in each iteration, starting with the null set; a set $S^{(i)}$ that stores the subscript of the features still not included in the model (also in the ith iteration) and $\xi^{(i)}$, a variable that represents the residuals of the model. The residual in each iteration $\xi^{(i)}$ is calculated as $1 - \hat{Y^{(i)}}$, so high probabilities produce low residual values. Iterations continue while the correlation distance between the residual and the input features are significant. In this case, $M^{(i)}$, $S^{(i)}$ and $\xi^{(i)}$ are updated, except when there is no improvement in the classification metric, that is, when the classification error does not decrease. In this particular situation $M^{(i)}$ and the metric used to evaluate the performance of the classifier are not updated. The main differences with the algorithm in Febrero-Bande and González-Manteiga (2019) is how the residuals are defined (in a regression context the residuals were defined as $\xi = Y - \hat{Y}$) and the fact that the metric for classification is different to the metric for a regression.

Algorithm 1 Variable selection using DC

Initialize:
$M^{(0)} = \emptyset, S^{(0)} = \{1, 2, ..., p\}, \xi^{(0)} = 1 - Y$
for $i = 1, ..., p$ **do**
   $\alpha^{(i)} = \underset{j \in S^{(i-1)}}{\mathrm{argmax}} \ \mathcal{R}(X_j, \xi^{(i-1)})$
  if $\mathcal{R}(X_{\alpha^{(i)}}, \xi^{(i-1)})$ significant, **then**
     $M^{(i)} = M^{(i-1)} \cup X_{\alpha^{(i)}}$
     $S^{(i)} = S^{(i-1)} \backslash \left\{ s^{(i)} \right\}$
     $\hat{Y}^{(i)} = f(M^{(i)})$
     $\xi^{(i)} = 1 - \hat{Y}^{(i)}$
     if $CE^{(i)} > CE^{(i-1)}$ **then**
       $M^{(i)} = M^{(i-1)}$
       $IoU^{(i)} = IoU^{(i-1)}$
     end if
  else
    Stop
  end if
end for

The metric used to measure the correlation between residuals and features (vector or functional covariates) is distance correlation $\mathcal{R}(X, Y)$ (Székely et al. 2007; Székely and Rizzo 2014). It is defined as follows:

$$\mathcal{R}(X, Y) = \frac{d\mathrm{Cov}^2(X, Y)}{\sqrt{d\mathrm{Var}^2(X) d\mathrm{Var}^2(Y)}} \tag{10}$$

where distance covariance, $d\text{Cov}^2(X, Y)$, and distance variance, $d\text{Var}^2(\cdot)$, are doubly centered Euclidean distances among all the elements of the X and Y.

This metric fulfills two important conditions that differentiate it from the Pearson's correlation:

- $\mathcal{R}(X, Y)$ is defined for X and Y random vector variables in arbitrary, not necessarily equal, finite dimension spaces.

- $\mathcal{R}(X, Y) = 0$ characterizes the independence of X and Y, even when the independence is nonlinear.

Therefore, distance correlation is able to detect not only linear but also nonlinear dependence between two variables. Distance correlation satisfies $0 \leq \mathcal{R} \leq 1$.

3. TESTING WITH ARTIFICIAL DATA

3.1. DATA GENERATION

To assess the situation when the variables are of different natures, we have developed a simulation study to check the performance of the algorithm in a mixed scenario with functional and scalar variables.

Five functional and five scalar variables were simulated, and the response was constructed as a function of the first two functional and the first two scalar variables. The functional variables, $\mathcal{X}_1, \ldots, \mathcal{X}_5$, were generated following Ornstein–Uhlenbeck processes in [0, 1] independently of each other. The scalar variables Z_1 and Z_5 follow a distribution U[0, 1], while Z_2, Z_3 and Z_4, follow a N(0, 1). So, in order to check how the procedure selects covariates when they have different natures we constructed the response as follows:

$$\mathcal{Y} = 10 + a_1 \mathcal{X}_1 \beta_1 + a_2 \mathcal{X}_2^2 + 3a_3 Z_1 + a_4 Z_2^2 + \varepsilon \tag{11}$$

with $\beta_1 = 2t + \sin 4\pi t + 0.1, t \in [0, 1]$ and $\varepsilon \sim N(0, .25^2)$.

The coefficients $a = \{a_1, a_2, a_3, a_4\}$ were introduced to emphasize each part of the model in the following scenarios:

i. Functional linear effect: $a = \{1, 0, 0, 0\}$

ii. Functional with linear and nonlinear effects: $a = \{1, 1, 0, 0\}$

iii. Functional and scalar linear effect: $a = \{1, 0, 1, 0\}$

iv. Functional and scalar with nonlinear effect: $a = \{0, 1, 0, 1\}$

v. Functional and scalar with linear and nonlinear effects: $a = \{1, 1, 1, 1\}$

We estimated the model through different functional classification models using the first four principal components as functional covariates. Specifically, the functional models used are: functional random forest (FRF), functional support vector machines (FSVM) and functional generalized linear model (FGLM). The suffix.VS indicates that Algorithm 1 was

Table 1. Percentage of times that each covariate was included in the model for $B = 100$ replications. Theoretically, the numbers in bold should be 100 and the rest 0

Model	a	$\mathcal{X}_1$	$\mathcal{X}_2$	$\mathcal{X}_3$	$\mathcal{X}_4$	$\mathcal{X}_5$	Z_1	Z_2	Z_3	Z_4	Z_5
FRF.VS	{1, 0, 0, 0}	**100**	7	9	3	7	1	9	4	7	5
	{1, 1, 0, 0}	**100**	**100**	6	4	4	4	2	4	9	1
	{1, 0, 1, 0}	**100**	3	5	6	5	**100**	6	8	5	3
	{0, 1, 0, 1}	2	**100**	0	2	2	1	**100**	1	3	4
	{1, 1, 1, 1}	**100**	92	2	3	5	94	98	2	3	4
FSVM.VS	{1, 0, 0, 0}	**100**	3	4	3	2	4	4	3	0	1
	{1, 1, 0, 0}	**100**	**100**	2	2	1	5	7	3	4	1
	{1, 0, 1, 0}	**100**	3	6	2	1	**100**	2	6	2	0
	{0, 1, 0, 1}	3	**100**	1	1	1	6	**100**	9	1	1
	{1, 1, 1, 1}	99	95	1	1	2	90	98	3	4	5
FGLM.VS	{1, 0, 0, 0}	**100**	3	4	4	5	1	3	1	0	2
	{1, 1, 0, 0}	**100**	28	3	1	0	1	7	2	4	1
	{1, 0, 1, 0}	**100**	3	3	5	3	**100**	4	7	2	3
	{0, 1, 0, 1}	1	25	0	1	1	1	**76**	1	0	0
	{1, 1, 1, 1}	99	33	2	2	1	93	35	3	3	5

Bold values represent maximum values

used to select the variables. $n = 200$ samples were generated, and the process was repeated $B = 100$ times to stablish the percentage of times that a particular covariate enters the model.

The response was categorized in two different ways:

- Binary response model (Y_2): $\mathcal{Y}$ in (11) was categorized in two levels using the median as threshold: $Y_2 = 0$ if $\mathcal{Y} \leq q_{0.5}$, $Y_2 = 1$ if $\mathcal{Y} > q_{0.5}$, q_α being the quantile of order α of $\mathcal{Y}$ distribution

- Multinomial response model (Y_3): $\mathcal{Y}$ was categorized in three levels according to the following rule: $Y_3 = 0$ if $\mathcal{Y} \leq q_{0.33}$, $Y_3 = 1$ if $q_{0.33} < \mathcal{Y} \leq q_{0.67}$ y $Y_3 = 2$ if $\mathcal{Y} > q_{0.67}$.

3.2. NUMERICAL RESULTS

Table 1 shows the results obtained for different scenarios for the binary response. The suffix.VS indicates that Algorithm 1 was used to select the variables. As can be seen, the results are very good. $\mathcal{X}_1$ was selected almost all the time in the 100 repetitions. For each repetition, the size of the test sample was 100. The non-relevant variables enter the model less than 10% of the time. GLM with feature selection, FGLM.VS, has problems to select the features with a nonlinear effect: $\mathcal{X}_2$ and Z_2 (scenarios ii, iv and v).

Table 2 shows the results for different combinations of a_1, a_2, a_3, a_4, for the multinomial response. Note that although the classification problem is slightly more complicated than for the binary response, the results are very good. Again, FGLM had difficulties selecting variables with a nonlinear effect: $\mathcal{X}_2$ y Z_2 (scenarios ii, iv and v).

Table 3 shows predictions, particularly we used intersection-over-union (IoU), also known as the Jaccard index, a metric widely used in semantic segmentation of images

Table 2. Percentage of times that a specific covariate was included in the model for $B = 100$ replications. Multinomial response with 3 classes, $\mathcal{Y}_3$

Model	a	$\mathcal{X}_1$	$\mathcal{X}_2$	$\mathcal{X}_3$	$\mathcal{X}_4$	$\mathcal{X}_5$	Z_1	Z_2	Z_3	Z_4	Z_5
FRF.VS	{1, 0, 0, 0}	**100**	5	3	7	8	6	9	12	6	5
	{1, 1, 0, 0}	**100**	**100**	6	6	5	4	4	4	4	3
	{1, 0, 1, 0}	**100**	8	9	13	7	**100**	6	7	8	2
	{0, 1, 0, 1}	2	**100**	8	4	1	4	**100**	3	2	4
	{1, 1, 1, 1}	**100**	88	6	3	7	96	95	4	5	5
FSVM.VS	{1, 0, 0, 0}	**100**	6	2	4	2	5	7	4	1	1
	{1, 1, 0, 0}	**100**	99	2	1	3	3	3	5	6	5
	{1, 0, 1, 0}	**100**	2	1	5	1	**100**	1	2	0	3
	{0, 1, 0, 1}	2	98	3	0	2	1	**100**	3	4	5
	{1, 1, 1, 1}	**100**	87	3	1	3	85	95	3	5	6
FGLM.VS	{1, 0, 0, 0}	**100**	8	3	6	6	5	5	9	3	3
	{1, 1, 0, 0}	**100**	34	4	6	3	5	2	1	4	4
	{1, 0, 1, 0}	**100**	6	6	8	5	**100**	3	5	2	4
	{0, 1, 0, 1}	0	24	0	1	2	0	**79**	0	1	0
	{1, 1, 1, 1}	**100**	27	5	4	7	94	35	5	7	4

Bold values represent maximum values

Table 3. Mean (in $B = 100$ repetitions) IoU in a test sample using multivariate and functional models, two types of response (binary and multiclass) and five different scenarios

$\mathcal{Y}$	a	RF	FRF	FRF.VS	SVM	FSVM	FSVM.VS	GLM	FGLM	FGLM.VS
$\mathcal{Y}_2$	{1, 0, 0, 0}	0.75	0.79	0.84	0.74	0.74	0.82	0.78	0.85	**0.86**
	{1, 1, 0, 0}	0.75	0.79	0.84	0.74	0.74	0.82	0.78	0.85	**0.86**
	{1, 0, 1, 0}	0.75	0.77	0.84	0.74	0.76	0.81	0.82	0.87	**0.88**
	{0, 1, 0, 1}	0.63	0.67	**0.77**	0.52	0.48	0.73	0.42	0.42	0.42
	{1, 1, 1, 1}	0.60	0.63	**0.67**	0.58	0.55	0.66	0.50	0.52	0.52
	Mean	0.67	0.70	**0.76**	0.64	0.62	0.74	0.61	0.64	0.65
$\mathcal{Y}_3$	{1, 0, 0, 0}	0.57	0.63	0.68	0.57	0.58	0.69	0.64	0.73	**0.76**
	{1, 1, 0, 0}	0.41	0.45	**0.53**	0.44	0.39	0.49	0.34	0.36	0.37
	{1, 0, 1, 0}	0.53	0.57	0.69	0.54	0.59	0.66	0.68	0.77	**0.80**
	{0, 1, 0, 1}	0.38	0.43	**0.58**	0.32	0.29	0.52	0.25	0.25	0.25
	{1, 1, 1, 1}	0.30	0.43	**0.49**	0.41	0.39	0.45	0.33	0.34	0.35
	Mean	0.46	0.50	**0.60**	0.46	0.45	0.56	0.45	0.49	0.51

Bold values represent maximum values

and point clouds. It should be noted that we have simulated scenarios that are difficult to classify; hence, the results would be better for simpler scenarios. An analysis of the table leads to the following conclusions:

- For both the binary and the multinomial responses, functional approaches outperform their non-functional counterparts

- Mean IoU is larger for a binary response than for a multinomial response

- Appropriate variable selection contributes to improve the results, although this depends on the classifier
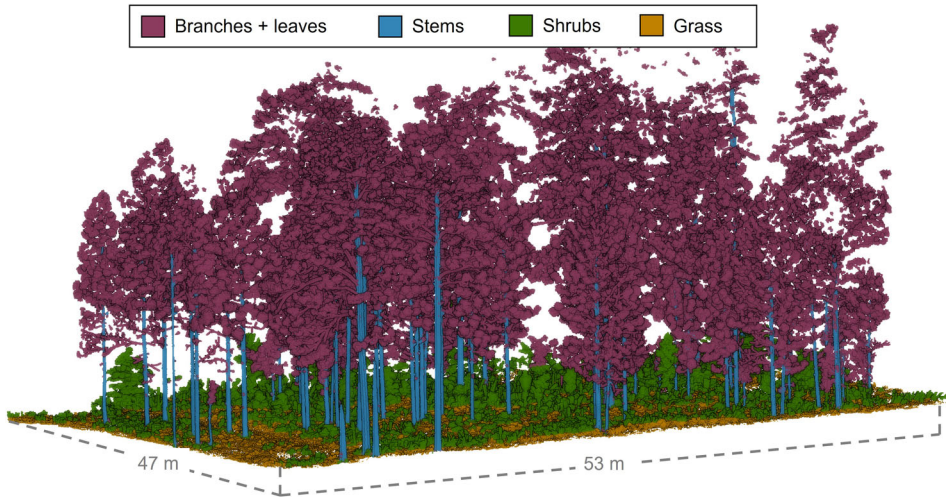
Figure 2. Terrestrial laser scanning point cloud from of a longleaf pine forest plot. Colors correspond to different vegetation structure categories that were manually classified.

- Among the functional models, the best results were in general obtained using FRF, while the GLM with penalization (it is a linear model) led to the worst results when there were nonlinear effects
- The maximum mean IoU corresponds to the functional approach with Functional Random Forest and variable selection.

4. CASE STUDY

4.1. DATASET AND FEATURE EXTRACTION

To test the classifications, we used a terrestrial laser scanning point cloud from a longleaf pine (Pinus palustris) plot in Pebble Hill, Georgia, USA. The plot dimensions are approximately 47×53 m ($2500\,\text{m}^2$). The scanner (RIEGL VZ2000) was placed in 8 positions throughout the plot, and the scanning density was set up so neighboring points on an ideal surface at 10 m from the sensor would be 6 mm apart. The point clouds from the 8 different scans were registered and joined using the software RisCAN Pro 2.0 (RisCAN Pro 2022), resulting in a total of 22 million points after removing duplicates within 5 mm of each other. The plot contained 83 trees (250 trees per ha), and it was completely covered by understorey vegetation (grasses and shrubs), with a horizontal shrub coverage of approximately 60%. The average diameter at breast height of the tree was 28 cm, and the dominant height was 22 m.

The point cloud was manually and visually classified into four classes as regards the vegetation structure: branches and leaves (69% of the points), stems (7%), shrubs (12%), and grasses (12%). Figure 2 shows the resulting manually classified point cloud.

In order to extract the features representing the local geometry around each point, the eigenvalues and eigenvectors of the covariance matrix constructed from the coordinates of

Table 4. Features extracted from the point cloud. $\mathbf{z}$ represents a vector in the direction of the vertical axis

Name	Formula
Eigenvalue sum	$\lambda_1 + \lambda_2 + \lambda_3$
PCA$_1$	$\frac{\lambda_1}{(\lambda_1+\lambda_2+\lambda_3)}$
PCA$_2$	$\frac{\lambda_2}{(\lambda_1+\lambda_2+\lambda_3)}$
N_x	$(\mathbf{v}_3)_x$
N_y	$(\mathbf{v}_3)_y$
N_z	$(\mathbf{v}_3)_z$
Linearity	$\frac{\lambda_1-\lambda_2}{\lambda_1}$
Planarity	$\frac{\lambda_2-\lambda_3}{\lambda_1}$
Anisotropy	$\frac{\lambda_1-\lambda_3}{\lambda_1}$
Sphericity	$\frac{\lambda_3}{\lambda_1}$
Omnivariance	$\sqrt[3]{\lambda_1\lambda_2\lambda_3}$
Eigenentropy	$\sum_{i=1}^{3} \lambda_i \ln\lambda_i$
Surface variation	$\frac{\lambda_3}{(\lambda_1+\lambda_2+\lambda_3)}$
Horizontality	$\frac{a\cos(\mathbf{v}_3\cdot\mathbf{z})}{\|\mathbf{v}_3\|}$

the points in a sphere of a specific radius were calculated. This was accomplished through the eigendecomposition of the covariance matrix Σ (Ordóñez and Cabo 2017; Thomas et al. 2018):

$$\Sigma = \frac{1}{N} \sum_{i=1}^{N}(\mathbf{p}_i - \bar{\mathbf{p}})^{\mathrm{T}}(\mathbf{p}_i - \bar{\mathbf{p}}) = \mathbf{V}\Lambda\mathbf{V}^{-1} \tag{12}$$

where N is the number of points in the sphere, $\mathbf{p}_i = (X_i, Y_i, Z_i)$ a vector with the coordinates of each point, $\mathbf{V}$ a matrix whose columns are the eigenvectors $\mathbf{v}_i, i = 1, .., 3$, and Λ a diagonal matrix whose nonzero elements are the eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$.

The three eigenvalues and the eigenvector $\mathbf{v}_3$ were used to calculate different features registered in Table 4. Most of these features include mathematical operations with the values of the eigenvalues which are linked to an ellipsoid that represents the local geometry around each. Thus, when $(\lambda_1 \gg \lambda2, \lambda3 \simeq 0)$ we face a linear structure. Similarly, $\lambda_1, \lambda_2 \gg \lambda_3 \simeq 0$ indicates a planar geometry, while $\lambda_1 \simeq \lambda_2 \simeq \lambda_3$ corresponds to a local volumetric geometry. A more detailed study of the geometrical meaning of these and other local features can be obtained in Demantké et al. (2011); Dittrich et al. (2017). In addition, the Z coordinate (elevation above ground) was also included as it is a discriminant variable, especially in order to distinguish grasses and shrubs from the crown. Obviously, this variable is independent of the scale.

The features defined in Table 4 were calculated at 60 different scales (i.e., 60 different search radii around each point), evenly spaced between 2.5 and 150 cm.
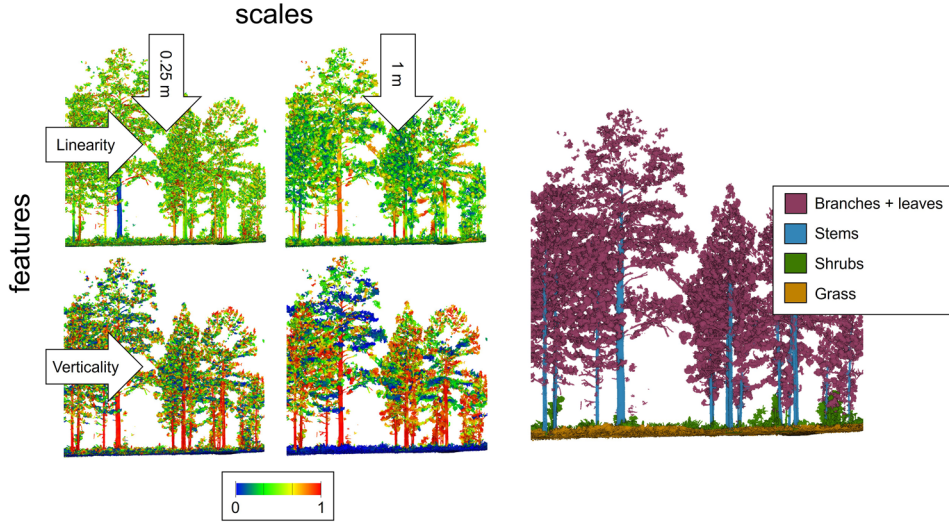
Figure 3. Subset of the study dataset with the manually classified point cloud and two features (Verticality and Linearity) computed at two different scales (0.25 and 1 m) .

4.2. Results and discussion

A subset of the test/training point cloud is shown in Fig. 3, along with the representation of two features (Verticality and Linearity) at two different scales (0.25 and 1 m). Clear differences can be seen for the points belonging to certain classes by using a single feature; for instance, tree trunks are clearly distinguishable from the Verticality associated with each point. However, it is also easy to see in Fig. 3 how the use of different scales can dramatically change the values for the same feature and same class (e.g., some large stems having low linearity values, in blue in the figure, at a 0.25 m scale, and high values, in red, at a 1 m scale).

Figure 4 is a heatmap of DC values for each pair of features as well as each of the features with the response variable, for all the scales. As can be seen, the response variable is more strongly correlated with verticality (DC = 0.39) than with the rest of the features. In general, the derivatives are less strongly correlated with the response than the original features. It can also be seen that some features are highly correlated with each other (i.e., Linearity has a high correlation with PCA1 and PCA2, and Sphericity is quite highly correlated with Surface variation). This explains why the algorithm we propose here for feature selection does not incorporate many of the features in the final model.

Figure 5 shows the mean value curves for six of the features (Verticality, Anisotropy, Surface variation, Linearity, Planarity and Sphericity), for all the data in the training sample, and for each category. These features were, in the same order, the most important according to the feature selection algorithm used. Figure 5 makes it clear that there are some differences among the features for each category. For instance, the mean values for Verticality are, as would be expected, very different at all the scales for points on stems and grass, and very similar for points on shrubs and branches-leaves at most scales. Also, in some features (e.g., Anisotropy, Linearity and Planarity), points on stems show very sudden changes at
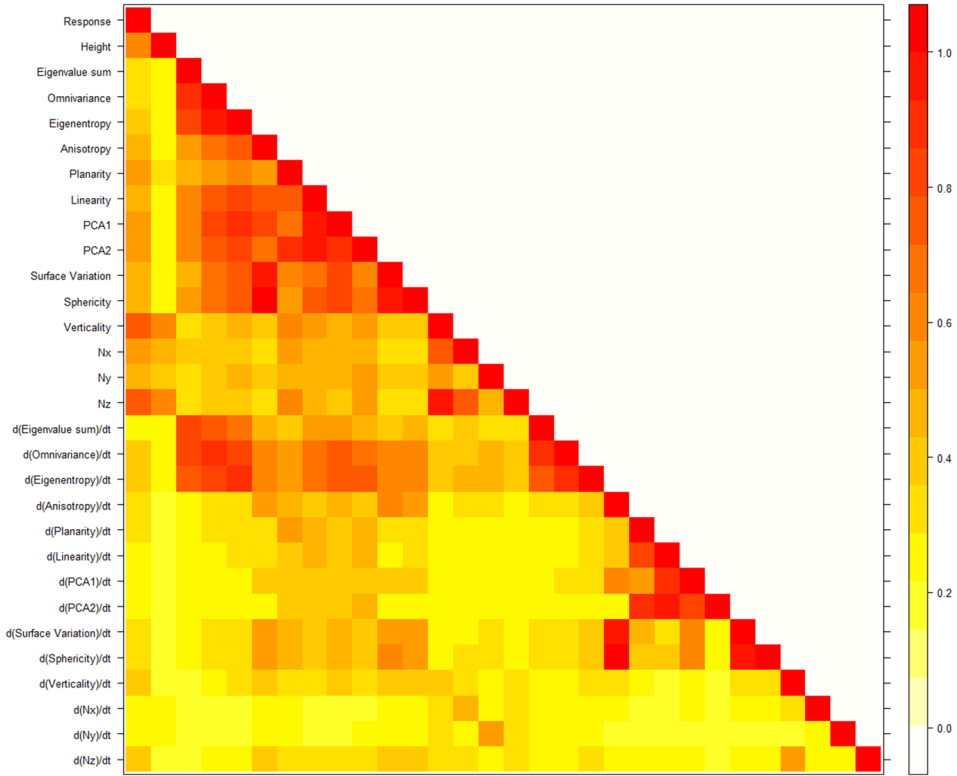
Figure 4. Heatmap of distance correlation values for all the features and the response variable.

the smallest scales, but stabilize around 30 cm, which is close to the average stem diameter. In general, the mean value curves show a clear stabilization pattern for all the features at scales that are above 30 or 40 cm. Note that in very dense point clouds, like that of the test forest plot, the use of scales (i.e., search radii around each point for feature computations) larger than 50–70 cm implies that a large number of points from different classes are likely to be included. This could be considered 'contamination' of the feature calculations, as the computation is not performed with points from the same class.

Table 5 shows the metrics used to evaluate the different models: IoU for each class and mean IoU of all the classes. The models used were SVM, GLM and RF. The results are shown for both, multivariate and functional models (Cabo et al. 2019). For the latter, two different functional approaches were used: smoothing with B-Splines (BSP) and using principal components instead of the original features (FPCA). In addition, for these two functional approaches we also analyzed the effect of incorporating to the model the first derivatives and the four most frequently selected features in all the variable selection processes: Verticality, Anisotropy, Surface variation, and elevation above the ground (Z coordinate). Furthermore, two popular deep learning models, specifically PointNet and DGCNN, were tested. These two models use the coordinates of the points and the components of the normals at $k = 0.5$ meters as input features. A total of 6,000,000 points were used to train the DL models, many
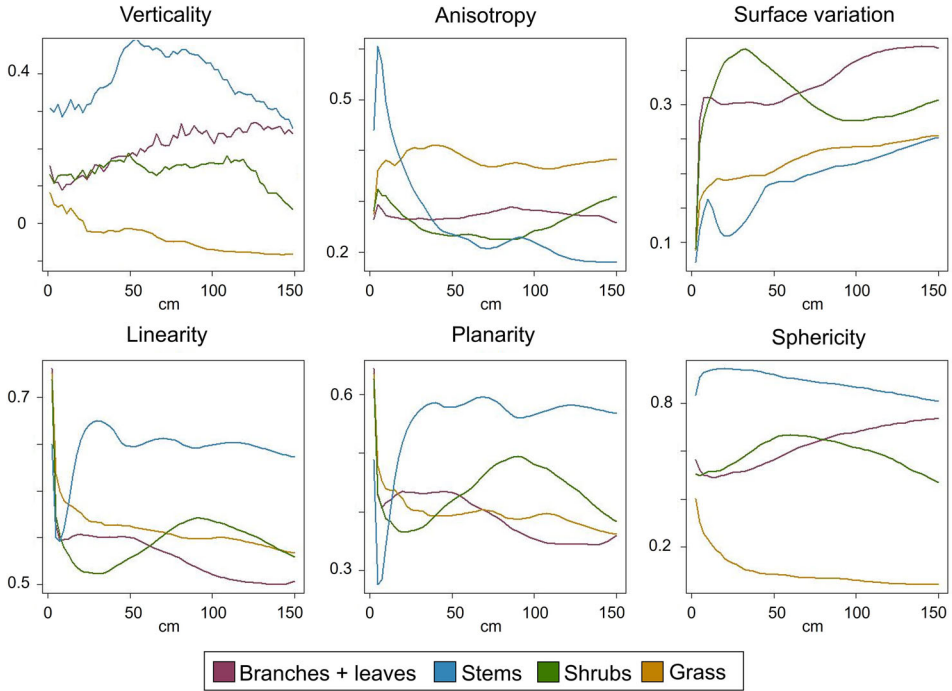
Figure 5. Mean value curves for six of the features (Verticality, Anisotropy, Surface variation, Linearity, Planarity and Sphericity), for all the data in the training sample, and for each category.

more than the 10,000 points used for the ML models. Larger sample sizes did not produce better results.

The multivariate models were trained considering all the features and scales and also after applying an algorithm of recursive feature elimination (Kuhn 2016). As usual, different models provide different results, and in this particular case SVM and RF slightly outperformed GLM. It is also observed that the results improve after applying feature selection. The functional approach does not produce a substantive improvement with respect to the multivariate analysis even when the derivatives of the features are included in the model. There is also no significant difference between using principal components (FPCA) or splines (BSP). However, the use of the four-variable selection clearly improves the results, but even in this case using FPCA instead of BSP does not improve the results, and indeed worsens them. Comparison with deep learning methods was uneven, as our best model (BSP with variable selection of functional features) achieves a higher mean IoU value than PointNet but lower than DGCNN.

Regarding the classification performance in the different classes, as shown in Table 5 there is neither a clear and definite pattern nor any one class with a clearly better or worse performance. However, in the best classification models in terms of IoU (those using a four-variable selection), branches and leaves seem to have slightly better results than the other classes. Also, in the multivariate approach, in all the models (SVM, GLM and RF), points on shrubs showed clearly worse results than the other classes, which is probably due to some confusion in distinguishing between branches-leaves and shrubs, which is reduced with the

Table 5. Intersection over union (IoU) for each class and average values using SVM, GLM, Random Forest (RF) and two deep learning models: PointNet and DGCNN.

GLM		feature selection			derivatives		variable selection	
IoU	GLM	GLM	FPCA	BSP	FPCA	BSP	FPCA	BSP
mean IoU	0.535	0.584	0.571	0.591	0.577	0.553	0.648	**0.694**
branches + leaves	0.605	0.617	0.626	0.599	0.619	0.543	0.738	**0.781**
grass	0.583	0.696	0.684	0.733	0.711	0.747	0.734	**0.720**
shrubs	0.463	0.559	0.518	0.537	0.51	0.436	0.584	**0.629**
stems	0.488	0.466	0.457	0.494	0.466	0.485	0.535	**0.646**
RF		feature selection			derivatives		variable selection	
IoU	RF	RF	FPCA	BSP	FPCA	BSP	FPCA	BSP
mean IoU	0.583	0.706	0.708	0.710	0.719	0.733	0.768	**0.773**
branches + leaves	0.557	0.716	0.755	0.775	0.759	0.769	0.803	**0.806**
grass	0.652	0.752	**0.799**	0.769	0.774	0.780	0.764	0.766
shrubs	0.451	0.662	0.632	0.602	0.651	0.652	0.728	**0.736**
stems	0.673	0.0.692	0.645	0.693	0.690	0.730	0.776	**0.782**
SVM		feature selection			derivatives		variable selection	
IoU	SVM	SVM	FPCA	BSP	FPCA	BSP	FPCA	BSP
mean IoU	0.544	0.594	0.692	0.730	0.701	0.721	0.727	**0.766**
branches + leaves	0.619	0.629	0.697	0.711	0.696	0.745	0.747	**0.804**
grass	0.574	0.709	0.753	0.750	**0.769**	0.744	0.737	0.710
shrubs	0.476	0.556	0.686	0.687	0.684	0.634	0.709	**0.727**
stems	0.508	0.482	0.630	0.771	0.654	0.762	0.716	**0.823**

DL		
IoU	PointNet	DGCNN
mean IoU	0.668	**0.794**
branches + leaves	**0.979**	0.971
grass	0.344	**0.759**
shrubs	0.543	**0.646**
stems	**0.806**	0.801

Bold values represent maximum values

Within each machine learning model, IoU values are shown for all the options considered: standard multivariate approaches and functional approaches: functional principal component analysis (FPCA), B-splines (BSP), their derivatives, and the result of a variable selection (Verticality, Anisotropy, Surface variation, and elevation above the ground)

functional approaches. DGCNN model provided very good results in all the categories with the exception of shrubs. For its part, PointNet showed good results for stems and branches but very bad results for grasses and shrubs.

Some differences between the solutions provided by the two DL models and the RF(BSP + VS) functional approach are showed in Fig. 6. The functional model classifies as branches an leaves some points located on top of the stems. It is also visible that the functional model classifies groups of points corresponding to shrub as branches and leaves. On their part, the two DL models tend to classify as stem groups of points that should be classified as branches and leaves.
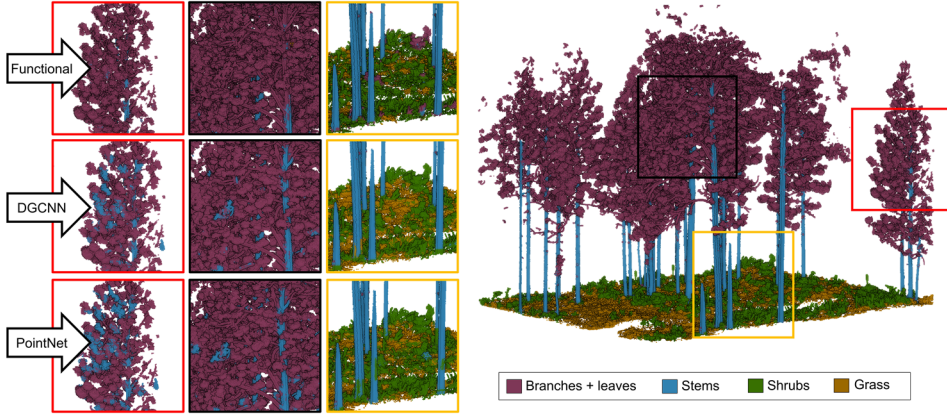
Figure 6.   Point cloud used to evaluate the performance of the mathematical models. The rectangles of each color delimit the enlarged areas in the general view of the plot (right). Detail of the classification carried out by the two DL models and the functional model RF (BSP + VS) in the areas delimited by the rectangles (left).

Table 6.   Computing time in seconds for the best multivariate and functional models (in terms of mean IoU) and for the two deep learning models tested

Model	N_{train}	CPU	GPU	N_{test}	CPU	GPU
RF (FS)	10,000	9.2		6,829,216	253.1	
RF(BSP)	10,000	10.6		6,829,216	1,611.4	
RF(BSP+VS)	10,000	99.7		6,829,216	435.5	
PointNet	6,000,000	24,376.1	661.2	6,829,216	171.7	6.5
DGCNN	6,000,000	61,434.0	751.6	6,829,216	81.10	9.2

For the two deep learning models, a distinction is made between CPU and GPU processing. N_{train} and N_{test} are the sizes of the training and test samples, respectively

Table 6 shows the computing time (in seconds) during training for the multivariate and functional approaches with the highest mean IoU, as well as for the two DL models tested. All the models were run in a computer with Windows 11 and the following features: Intel(R) Core(TM) i7-8550U CPU, 1.80 GHz, 16.0 GB RAM. The two DL models were also training using a GPU Geoforce RTX 2080. As can be seen, training DL models require much more computation time than training ML or functional models to reach comparable results, except when the proposed feature selection method is applied to the functional features. This is because of the calculation of the distance correlation in each iteration. Accordingly, if we look for a balance between classification error and computing time, selecting variables following our method would not be the best option. However, selecting the important features has some advantages such as improving the interpretability of a model or reducing the negative effects of collinearity or concurvity. The time difference reduces considerably when a GPU is used to train the DL models.

The size of the training sample in Table 6 is different for DL and ML models for two reasons. On the one hand, DL models do not generalize well for samples as small as 10,000 points, so it is not useful to use such a small sample with the DL algorithm. On the other hand,

ML models are computationally expensive for large sample sizes above that value. However, this is not a major drawback as we have verified that for larger samples up to 20,000 points the results of the ML models do not improve significantly. Regarding inference times (sample test), execution time is longer for ML models than for DL models (for the same sample size) specially in the case of the functional data model with variable selection RF(BSP+VS).

5. CONCLUSIONS

In this work, we illustrate the use of functional data analysis for multiscale semantic segmentation of 3D point clouds as an alternative to the standard multivariate analysis. The functional data analysis approach avoids the problems associated with not considering relevant scales or having to search for them, which is a drawback in the standard approach. We compare different adaptations of the multivariate models to the functional case, which offers a balanced compromise between predictive capacity and simplicity.

The results obtained using artificial data concur with the initial hypothesis that approaching the features by functions and selecting some features result in a better approximation to the data. All the finally selected models combined functional and scalar information successfully, except for the functional GLM when the relationship between the variables is nonlinear. In general, the application of the proposed feature selection algorithm improves the accuracy of the model.

In terms of the application to real data, there is no significant improvement of the results when features are approximated by functions, except when a process of variable selection is applied. Finally, however, the best model only included four of the fourteen initial variables. Of the four categories studied, the one including branches and leaves was slightly better classified than the other three. Conversely, shrubs were the worst classified category due to their confusion with other classes, especially with grass, although the functional approach mitigates this effect.

The comparison of our approach with two deep learning models for 3D point cloud semantic segmentation was promising. Considering the mean IoU as the metric for comparison, functional models with feature selection outperformed PointNet, but DGCNN surpassed all the ML and functional models. When the IoU for each category is compared, branches + leaves and stems were mainly better classified with the DL algorithms, but not the other two categories. DL models faster in inference than ML models. If other aspects such as interpretability, training sample size or training computation time are taken into account, the functional approach can be considered superior to those based on deep learning, with exception of the method with feature selection that it is very time consuming.

SUPPLEMENTARY MATERIALS: CODES AND DATA

The codes and data used in this paper are available in the GitHub repository https://github.com/moviedo5/FDA_3D_Point_Cloud/. From the pkg folder, the package fda.usc.devel (devel version of fda.usc, Febrero-Bande et al. (2012)) can be

installed. From the `pkg` folder, the package `fda.usc.devel` (devel version of `fda.usc`, Febrero-Bande et al. (2012)) can be installed an essential requirement for reproducing the provided examples.

ACKNOWLEDGEMENTS

Declarations
Conflict of interest The authors declare no conflict of interest in this article. The findings and conclusions in this publication are those of the authors and should not be construed to represent any official USDA or US Government determination or policy.

REFERENCES

Lee I, Schenk T (2002) Perceptual organization of 3D surface points. Int Arch Photogramm Remote Sens Spat Inf Sci 34(3/A):193–198

Linsen L, Prautzsch H (2001) Local versus global triangulations (2001) In: Proceedings of Eurographics 1: 257-263

Guo B, Huang X, Zhang F, Sohn G (2015) Classification of airborne laser scanning data using JointBoost. ISPRS J Photogramm Remote Sens 100:71–83

Cabo C, Ordóñez C, Sáchez-Lasheras F, Roca-Pardiñas J, de Cos-Juez J (2019) Multiscale supervised classification of point clouds with urban and forest applications. Sensors 19:4523

Xie Y, Tian J, Zhu XX (2020) Linking points with labels in 3D: a review of point cloud semantic segmentation. IEEE Geosci Remote Sens Mag 8:38–59

Brodu N, Lague D (2012) 3D terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: applications in geomorphology. ISPRS J Photogramm Remote Sens 68:121–134

Zhang J, Zhao X, Chen Z, Lu Z (2019) A review of deep learning-based semantic segmentation for point cloud. IEEE Access 7:179118–179133. https://doi.org/10.1109/ACCESS.2019.2958671

Qi R, Su H, Kaichun M, Guibas LJ (2017) PointNet: deep learning on point sets for 3D classification and segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), pp 652-660

Qi R, Yi l, Su H, Guibas LJ (2017) PointNet++: Deep hierarchical feature learning on point sets in a metric space. In: Advances in neural information processing systems (NIPS), pp 5105-5114

Wang Y, Sun Y, Liu Z, Sarma SE, Bronstein MM, Solomon JM (2018) Dynamic graph CNN for learning on point clouds. ACM Trans Graph. https://doi.org/10.1145/3326362

Tchapmi L, Choy C, Armeni I, Gwak J, Savarese S (2017) SEGCloud: semantic segmentation of 3D point clouds. 2017 International Conference on 3D Vision (3DV), pp 537-547, https://doi.org/10.1109/3DV.2017.00067.

Hu Q, Yang B, Xie L, Rosa S, Guo Y, Wang Z, Trigoni N, Markham A (2021) RandLA-Net: efficient semantic segmentation of large-scale point clouds. In: Proceedings of the IEEE conference on computer vision and pattern recognition

Zhu X, Zhou H, Wang T, Hong F, Ma Y, Li W, Li H, Lin D (2021) Cylindrical and asymmetrical 3D convolution networks for LiDAR segmentation. IEEE Trans Pattern Anal Mach Intell. https://doi.org/10.1109/TPAMI.2021.3098789

Zhao H, Jiang L, Jia J, Torr P, Koltun V (2021) Point transformer. IEEE/CVF Int Conf Comput Vision 2021:16239–16248. https://doi.org/10.1109/ICCV48922.2021.01595

Chang AX, Funkhouser T, Guibas L, Hanrahan P, Huang Q, Li Z, Savarese S, Savva M, Song S, Su H, Xiao j, Yi L, Yu F (2015) ShapeNet: an information-Rich 3D model repository. In: Technical report arXiv:1512.03012 [cs.GR],

Dai A, Chang AX, Savva M, Halber M, Funkhouser T, Nießner M (2017) Scannet: richly-annotated 3d reconstructions of indoor scenes. arXiv preprint arXiv:1702.04405, 2017

Behley J, Garbade M, Milioto A, Quenzel J, Behnke S, Stachniss C, Gall J (2019) Semantickitti: a dataset for semantic scene understanding of lidar sequences. Technical Report arXiv:1904.01416

Demantké J, Mallet C, David N, Vallet B (2011) Dimensionality based scale selection in 3D lidar point clouds. In: Laserscanning 2011, Calgary, Canada, hal-02384758f

Mitra NJ, Nguyen A (2023) Estimating surface normals in noisy point cloud data. Proc Ninet Ann Symp Comput Geom 2003:322–328

Weinmann M, Jutzi B, Hinz S, Mallet C (2015) Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. ISPRS J Photogramm Remote Sens 105:286–304

Shannon C (1948) A mathematical theory of communication. Bell Syst Tech J 27:379–423

Oviedo-de la Fuente M, Cabo C, Ordóñez C, Roca-Pardiñas J (2021) A distance correlation approach for optimum multiscale selection in 3d point cloud classification. Mathematics 9(12):1328. https://doi.org/10.3390/math9121328

Jović A, Brkić K, Bogunović N (2015) A review of feature selection methods with applications. In: 38th international convention on information and communication technology. Electronics and microelectronics (MIPRO) pp 1200–1205. https://doi.org/10.1109/MIPRO.2015.7160458

Febrero-Bande M, González-Manteiga W, Oviedo De La Fuente M (2019) Variable selection in functional additive regression models. Comput Stat 34:469–487. https://doi.org/10.1007/s00180-018-0844-5

Thomas H, Goulette F, Deschaud JE, Marcotegui B., LeGall Y (2018) Semantic classification of 3D point clouds with multiscale spherical neighborhoods. In: 2018 International conference on 3D vision (3DV). IEEE, pp 390-398

Atik ME, Duran Z, Seker DZ (2021) Machine learning-based supervised classification of point clouds using multiscale geometric features. ISPRS Int J Geo-Inf 10:187. https://doi.org/10.3390/ijgi10030187

Blum AL, Langley P (1997) Selection of relevant features and examples in machine learning. Artif Intell 97(1–2):245–271

Kuhn M, Johnson K (2019) Feature engineering and selection a practical approach for predictive models, 1st edn. Chapman & Hall/CRC Data Science Series, Boca Raton FL

Fraiman R, Gimenez Y, Svarcm M (2016) Feature selection for functional data. J Multivar Anal 146:191–208

Berrendero JR, Cuevas A, Torrecilla JL (2016) Variable selection in functional data classification: a maxima-hunting proposal. Stat Sin 26(2):619–638

Hastie T, Tibshirani R (1987) Generalized additive models: some applications. J Am Stat Assoc 82(398):371–386

Breiman L (2001) Random forests. Mach Learn 45(1):5–32

Möller A, Tutz G, Gertheiss J (2016) Random forests for functional covariates. J Chemom 30(12):715–721

Boser B, Guyon I, Vapnik V (1992) A training algorithm for optimal margin classifiers. In: COLT '92: Proceedings of the fifth annual workshop on computational learning theory, pp 144-152. https://doi.org/10.1145/130385.130401

Rossi F, Villa N (2006) Support vector machine for functional data classification. Neurcomputing 69:730–742

Friedman J, Hastie T, Tibshirani R (2010) Regularization paths for generalized linear models via coordinate descent. J Stat Softw 33(1):1–22

Hackel T, Wegner JD, Schindler K (2016) Fast semanticbsegmentation of 3D point clouds with strongly varying density. ISPRS Ann Photogramm Remote Sens Spat Inf Sci 3:177–184

Ramsay JO, Silverman BW (1997) Functional Data Analysis. Springer, New York

Han Lin Shang (2014) A survey of functional principal component analysis. AStA Adva Stat Anal 98:121–142

Florence N (2016) Functional principal component analysis of aircraft trajectories. [Research Report] RR/ENAC/2013/02, ENAC. 2013. ffhal-01349113ff

Székely GJ, Rizzo ML, Bakirov NK (2007) Measuring and testing dependence by correlation of distances. Ann Stat 35(6):2769–2794

Székely GJ, Rizzo ML (2014) Partial distance correlation with methods for dissimilarities. Ann Stat 42(6):2382–2412

http://www.riegl.com/products/software-packages/riscan-pro/. Last visited in April, 2022

Ordóñez C, Cabo C, Sanz-Ablanedo E (2017) Automatic detection and classification of pole-like objects for urban cartography using mobile laser scanning data. Sensors 17(7):1465

Dittrich A, Weinmann M, Hinz S (2017) Analytical and numerical investigations on the accuracy and robustness of geometric features extracted from 3D point cloud data. ISPRS J Photogramm Remote Sens 126:195–208

Kuhn M (2016) Contributions from Jed Wing, Steve Weston, Andre Williams, Chris Keefer, Allan Engelhardt, Tony Cooper, Zachary Mayer, Brenton Kenkel, the R Core Team, Michael Benesty, Reynald Lescarbeau, Andrew Ziem, Luca Scrucca, Yuan Tang and Can Candan. caret: Classification and Regression Training. R package version 6.0-71. https://CRAN.R-project.org/package=caret

Febrero-Bande M, Oviedo de la Fuente M (2012) Statistical computing in functional data analysis: the R package fda. usc. J Stat Softw 51(4):1–28. https://doi.org/10.18637/jss.v051.i04