Machine learning method for As-Is tunnel information model reconstruction

Machine Learning Method for As-Is Tunnel Information Model Reconstruction.

Nicola Rimella[a*], Lorenzo Rimella[b], Anna Osello[a]

[a]*DISEG, Politecnico di Torino, Corso Duca degli Abruzzi, 24, 10129 Turin, Italy*

[b] *ESOMAS, Universitá degli studi di Torino and Collegio Carlo Alberto, Piazza Vincenzo Arbarello, 8, 10122, Turin, Italy*

[*]*Corresponding author: nicola.rimella@polito.it*

Abstract

The maintenance of aging infrastructure requires advanced tools for efficient inspection and planning. This paper presents a methodology for segmenting and classifying point clouds of road tunnels to streamline maintenance operations. Processing large datasets, such as those generated by laser surveys, poses significant challenges without appropriate IT solutions. Data from four Italian tunnels were divided into segments and clustered into homogeneous groups. These clusters were manually classified to create a labeled dataset for training machine learning algorithms. The classified points, representing elements such as lining, road surfaces, and equipment, were then used to generate a 3D mesh model. By sharing the results in the OpenBIM format, the method facilitates seamless data exchange among infrastructure maintenance professionals, improving efficiency in planning and execution.

Keywords: Machine Learning-Clustering and Classification, Point Cloud, 3D reconstruction, Maintenance, Tunnelling, Building Information Model

1. Introduction

As described in [1], the status of Italian tunnels in the Trans-European Network (TERN) is difficult to manage. Approximately half of the total number of sections and lengths developments of TERN tunnels are in Italy and half of them have been in operation for more than 30 years. These structures need improvements, especially from a structural perspective. The same report [1] shows the data provided by the operators on the 31st of December 2020, revealing that only 18% of the TERN tunnels comply with the current regulations. The latest technologies, such as laser devices for collecting data on-site [2], can aid in describing complex structures, such as tunnels.

However, laser instruments produce an enormous amount of data that must be meticulously screened and cleared of unnecessary or unmanageable information before it can be used in the design process. To improve understanding of this big data, machine learning techniques can be used to support data screening [3]. A large-scale survey and investigation campaign has been undertaken in Italy to monitor the condition of underground infrastructure. The current investigation practice follows the guidelines in [4], using a layered approach to evaluate and understand the infrastructure. Level 0 involves taking a census of the structures in the territory that require renovation, including data on their geographical locations, maintenance histories, and other relevant details. Level 1 focuses on initial inspections and the processing of defect records. During this phase, it is also essential to gather information on the volume of the subsurface, identify visible defects through surveys, and perform geometric surveys. Level 2 involves analyzing the collected data and assigning an attention class to the infrastructure to evaluate its condition and determine if future

investigations and monitoring are needed. Level 3 provides an in-depth research campaign to estimate the instabilities related to the interaction of the lining with the natural formations it crosses. At this level, a point cloud of the tunnel is created. Level 4 provides accurate safety evaluations of the tunnel, based on the active danger and external environment factor. Lastly, Level 5 evaluates the importance of the infrastructure in the road network and the socio-economic context related to a hypothetical service disruption.

Table 1 Summary of aims during the multi-level approach survey.

Level	Aims
0	Geolocation - Census - Data collection
1	Identification of the state of preservation through preliminary and expeditive inspections
2	Estimation of attention class
3	Local surveys through depth inspections
4	Accurate global assessment
5	Only in the case of vulnerable assets, to be evaluated case-by-case

The current practice for collecting geometric data involves the use of ground-based laser scanners, to accurately reconstruct the tunnel morphology [2]. If the project data at Level 0 are not reliable or insufficient to fully understand the tunnel's lining, the laser scanner must be used at Level 1, creating an early-stage point cloud.

The use of OpenBIM technologies and methodologies [5] can provide an important contribution to the management of infrastructure inspections and maintenance by using shared and defined standards and improving collaboration across stakeholders. The current workflow leading to the definition of structural geometry, and the subsequent association of attention classes, is time-consuming and often does not leverage the highest accuracy of data acquired on-site. To achieve this work quickly yet meticulously, it is essential to organize the workflow and incorporate state-of-the-art computer tools that can handle data and translate it into formats that are compatible with the current software in use.

The methodology presented in this contribution aims to provide a scalable tool to support design studies, allowing significant time savings when processing large datasets from laser surveys. It also provides a more reliable and user-friendly tool for reconstructing BIM models describing the structure's geometry. The usage of the BIM model for operation and maintenance (O&M) of infrastructural assets is fundamental both for performing real-time analysis and storing data for future reference. In [6] the authors explain the importance of the use of BIM for the creation of a Digital Twin (DT) [7] that can be used for O&M. The accuracy of the BIM geometry can allow analysis of the structure's development and easily compare the actual state with the past history. This methodology is designed to be applied in a typical workflow of an engineering firm involved in tunnel maintenance and construction management, to analyze big data from laser surveys more efficiently and ensure high accuracy in geometry rendering.

Firstly, this paper describes the current methodology that is applied by most engineering firms to recreate a tunnel 3D model starting from a point cloud. Secondly, it reviews the technologies relevant to the proposed algorithm, defining and discussing their applications in infrastructure and tunneling. Thirdly, the methodology and algorithm developed to produce a Building Information Model (BIM) [8] using clustering and classification methods is presented. Application to four tunnels located in Italy will be presented along with a discussion on speed, accuracy and reliability. Due to the sensitivity of the infrastructure data, the raw dataset of the point cloud cannot be disclosed. However, we provide a GitHub repository [9] to easily reproduce our classification workflow on a cleaned and anonymized version of the data. Finally, conclusions and future developments of the work will be presented,

highlighting the limitations and potential of the presented methodology.

2. Current methodology and related works

The first step in understanding the morphology of a building or infrastructure is the accurate survey of its geometric parameters. Terrestrial laser scanning (TLS) is a widely used technology for three-dimensional data acquisition and mapping. Its use is well-established in tunnel investigation, for example, as shown in [10], these surveys could be used for deformation analysis and tunnel section extraction.

Laser scanner surveys are also essential to correctly plan maintenance work on tunnels when extraordinary interventions are required, such as the renovation of the structure.

Currently, to reconstruct the tunnel model from laser scanner surveys, cross-sections are made along a path that is similar to the tunnel alignment. From the intersection between a plane crossing the cloud and the point cloud itself, it is possible to extract curves that approximate the cross-section of the tunnel and to design a geometric model of the canopy's pattern by joining these cross-sections along the tunnel alignment. In [11] and [12] the authors describe the incredibly time-consuming process of manually purging noise points; however, this operation is crucial for the correct geometry reconstruction. A comparable technique is explained in [13], where a filtering algorithm is proposed to remove points that are not useful for the reconstruction of the section. These methods use only a small fraction of the data from the point cloud, focusing on those used to construct the sections, resulting in a significant loss of collected information. The use of a larger number of points, properly selected, allows to ensure an excellent congruence with the real geometry of the tunnel along its entire length.

Research regarding the application of machine learning techniques in tunneling is very active, and there are several examples showing the potential of these tools. [14] discusses the application of the RANSAC algorithm [15] for automatically cleaning cross-sections from noise points by interpolating a cylindrical geometry. This application of the algorithm gives high-quality results, but it is not easily scalable to tunnels presenting non-perfectly cylindrical shapes or with obvious deformations.

In [16] a multi-level approach for the clustering of tunnels from point clouds via RANSAC and DBSCAN [17] algorithms are considered. Here the algorithms are operated to cluster the elements directly by checking the correspondence with the parameters assigned to the algorithms, resulting in a significant automation of the procedure. However, this strategy is hard to scale to non-homogeneous density point clouds, as well as to be implemented with new classes of objects that may need to be identified. Provided with a fully classified dataset, where the clusters are correctly identified and labeled, one can adopt a supervised learning approach and train classification algorithms from the machine learning literature that can automate future cluster classification [18]. Examples of these algorithms span from basic logistic regression to neural networks and support vector machines [18].

In [19], the authors present an interesting application of algorithms for classifying prefabricated building elements. By using Industry Foundation Classes (IFC) methodology, the complex analyses carried out by these algorithms can be translated into more user-friendly data for workers who operate on-site, a key factor in successfully managing building and infrastructure data. In [20] an approach based on BIM and IFC format is presented to manage tunnel construction data, improving data exchange between project stakeholders. However, geometries derived from point clouds can be cumbersome to load into an infrastructure design or a game engine software. The problem is comparable to the one addressed by [21], where the authors reconstruct a 3D model of a baroque altarpiece from survey data. Due to the complexity of the architectural element and the number of points needed for accurate reconstruction, re-meshing steps are required to optimize it for software

integration. The proposed methodology aims to use the data from the survey to generate an IFC file describing the tunnel geometries and thus enable its use for design and execution work.

From a Machine Learning perspective, it is also important to draw a connection with the field of computer vision and specifically with segmentation methods. Supervised segmentation methods classify points from the 3D point cloud based on initial labeling via deep learning algorithms [22][23][24][25]. Our approach differs, as we label each cluster along with covariates representing summary statistics (e.g. number of points, volume, centroid of the cluster). There are several advantages to our tabular data approach compared to vanilla segmentation methods. Firstly, it offers computational efficiency from a memory perspective, as storing clusters instead of individual points reduces dataset size from the number of points times three (i.e., the x, y, z coordinates) to the number of clusters times the covariates. In our specific application, we found that storing only the summary statistics consumed just 0.1% of the original point cloud's memory. Secondly, companies are often reluctant to share point clouds due to data sensitivity. This is hampering the process of creating a highly heterogeneous dataset of tunnels which would be useful to train widely applicable classification algorithms. This privacy issue can be avoided by aggregating the point cloud in a cluster as per our procedure. Indeed, given a cluster and its covariates, it is not possible to reconstruct the point cloud, making the dataset fully anonymous. Thirdly, training of tree-based algorithms on tabular data runs almost instantaneously on most laptops. Moreover, these algorithms are incredibly light and can be easily integrated into portable devices. To conclude, we choose the most efficient and user-friendly solution, balancing computational cost, usability, and accuracy.

3. Methodology

The following section discusses the methodology used to classify tunnel elements and consequently enable As-Is modeling of the infrastructure. This approach is an alternative to the current practice based on cross-sections described in the previous section, to achieve a more reliable model in less time.

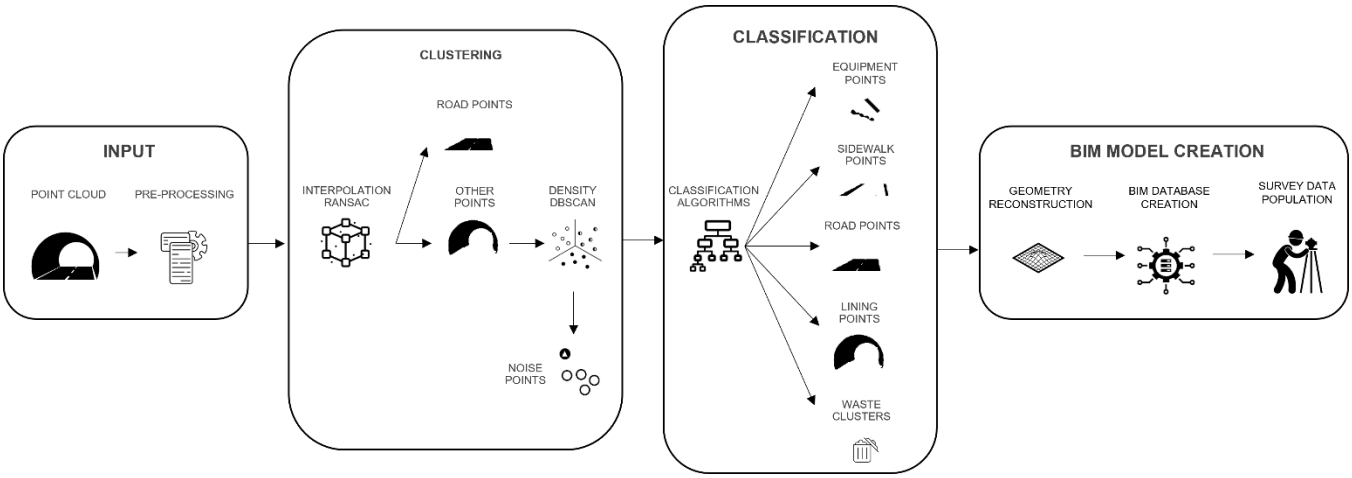

Figure 1. Methodological workflow. Starting from the left, boxes show how input data are clustered and then classified via classification algorithms. Finally, clean data are used to develop a BIM model.

Figure 1 shows the methodological process that was developed to create a BIM model from a point cloud. The diagram has been divided into four main blocks that display the algorithms and actions that are performed on the source data.

Starting from the left, "INPUT", the first block shows the input data, consisting of the point clouds derived from the laser scanner survey; an initial pre-processing activity is crucial to divide the dataset and organize the data, so as to optimize computation time.

The suitably arranged data then undergoes a clustering process in which the chosen algorithms clean, complete, split and segment the point cloud. For this intermediate step named "CLUSTERING", the RANSAC and DBSCAN algorithms are considered. The RANdom SAmple Consensus (RANSAC) [15] is an iterative algorithm for estimating the parameters of a mathematical model from a dataset containing outliers, which allows us to easily interpolate missing data in our tunnel. The subsequence clustering is performed with Density-Based Spatial Clustering of Applications with Noise (DBSCAN) [17] which distributes points into high density regions and automatically eliminates noise points, i.e. all those points that do not fit into the density groups provided. Here, both RANSAC and DBSCAN serve as purely empirical tools to segment the point cloud and make the labeling procedure faster, and nothing stops the user from considering other algorithms that fulfill the same purpose.

The different regions created during the "CLUSTERING" step are then manually labeled to equipment (E), sidewalk (S), road (R), lining (L) and waste (W), and then fed to "CLASSIFICATION" algorithms, see [26] and [27] for a review. The choice of the classification algorithm is not straightforward, and a careful selection must be performed. We suggest using the Python library "lazypredict" to test more than 20 algorithms on three different metrics (accuracy, balanced accuracy, F1-score). This should allow us to select the most performing candidates for subsequent fine-tuning of the hyperparameters.

After the "CLASSIFICATION" step five separate classes of clusters are produced, from which a reconstruction path leading to the development of an IFC model can follow. This is then used as a starting database for maintenance design and planning, representing the "BIM MODEL CREATION". As explained in Section 2, the creation of BIM models is currently a necessary procedure to design and manage the structural intervention on existing tunnels. Our work aims to improve the way point clouds are processed and the accuracy of the final BIM model.

4. Case study: four Italian tunnels

Section 4 demonstrates the application of this methodology for the segmentation, classification, and reconstruction of a BIM model for Italian tunnels. Subsection 4.1 discusses the tunnels used as data input. Subsection 4.2 describes the clustering algorithms used to segment the clouds and create the training dataset. The choice of the classification algorithm, data labeling, parameter tuning, and algorithm validation are described in Subsection 4.3. Subsection 4.4 presents the process for creating a simplified BIM model containing the meshes derived from the point cloud so that it can be used as the basis for the creation of a more complex parametric model.

4.1 Input

The data acquired by the TLS are multiple and depend on the used instrument. However, the choice of the tool, and consequently the collected data type, belongs to survey companies and not to planners. Therefore, it was decided to use, as starting input, point clouds that only contain point coordinate data: since all TLS tools acquire those data, this choice enables the process to be performed independently of the instrument used for the laser scanning, with differences depending only on the accuracy.

Data from four different tunnels are considered, showing differences in both geometry and survey tools used. These differences make it possible to build a heterogeneous dataset of tunnels that will provide multiple training examples for classification algorithms, making it as generalizable as the data allow.

Figure 2 shows the four different galleries used to train the algorithm. The first tunnel displayed in

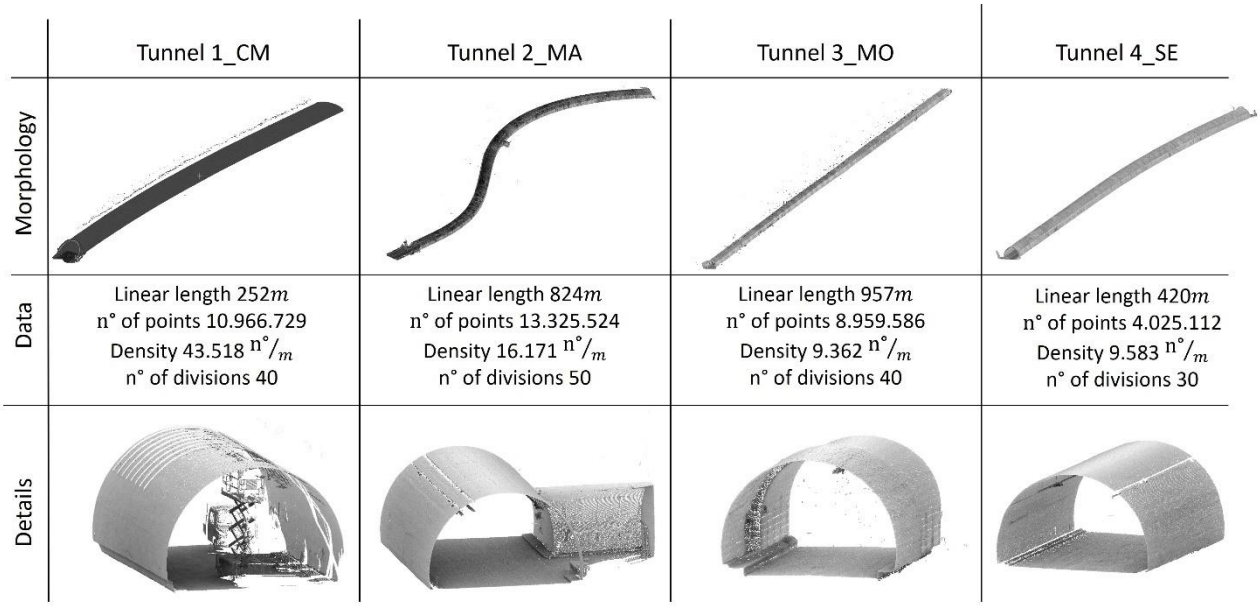

Figure 2 Four tunnels used for the experiments. From top to bottom: morphology, some data information, and details on some sections of the point clouds

the image, coded with 'CM', has the densest point cloud in the dataset, and, for most of its extension, all the tunnel parts are visible. On the inner intrados of the CM tunnel, there are also corrugated metal reinforcements, visible in the third row of Figure 2. However, in the first part of the tunnel, there are several disturbing elements such as machines, construction workers, traffic cones, and others. These disturbing components must be removed to correctly re-build the tunnel geometry, but their manual elimination is time-consuming. Identifying these elements by cross-section-based geometry reconstruction is extremely difficult: to make this easier, the developed methodology will allow these objects to be included in the classification process and quickly removed from the final geometry. The second tunnel, coded 'MA', was surveyed with less precision and has fewer points per linear meter of length than the CM tunnel. MA has a more curvilinear geometry than the other three tunnels and has a bypass, shown on the third row of *Figure 2*. These elements are not always present in tunnels, but their classification is essential to obtain a more flexible algorithm that adapts to different scenarios.

The third tunnel, coded with 'MO', and the fourth, called 'SE', has a similar cloud point density but different geometries. The morphology of MO is peculiar because it has sections that are narrowing down along the path and a significant amount of noise. From a morphological point of view, SE does not have any specific problems: however, some parts of the tunnel were not surveyed, such as the joining point between the lining and the pavement, visible in the third row of *Figure 2*.

The data described above must be properly pre-processed before being fed to clustering algorithms. It is then necessary to perform an initial division of the gallery into sub-parts not exceeding 300000 points, which allows the next processes to run in a few minutes. However, the segment of the tunnel analyzed cannot be too long, so sections with a similar curvature are considered. For each segment of the cloud, the direction of the normal vectors is also calculated as it will be used to compute some of the covariates to be used in the classification algorithm.

4.2 Clustering

The tunneling section, which has been previously set up as described in 4.1, can then undergo its first clustering process using the RANSAC algorithm for road identification. For this step of the clustering, the point cloud is interpolated with a plane for a certain number of iterations: the goal of

the RANSAC algorithm is to find the plane that maximizes the number of points within a chosen range. The choice of the hyperparameters, e.g. inlier/outlier ratio, of the RANSAC depends on the geometry of the tunnel and the quality of the survey. The roads in Italian tunnels can indeed be of different types and therefore need different hyperparameters to be properly separated from the rest of the cloud. The choice of the hyperparameters is then empirical and left to technicians for the best

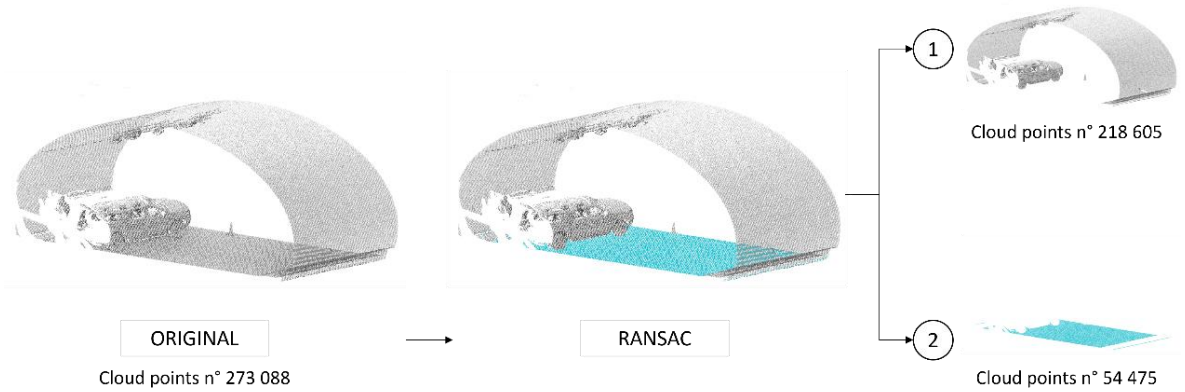

Figure 3. From left to right, input and output of the RANSAC algorithm. On the far right the separated clouds.

outcome.
In our application, we set the interaction parameter to 5000, to ensure accuracy in the results and avoid excessive computational time, and the inlier/outlier ratio to 2.5cm, to account for potential imperfections in the road surface.
Figure 3 shows the application of the RANSAC algorithm on a piece of the CM cloud, here the identified plane corresponds to the road surface of the tunnel as it is the only plane element with enough points to satisfy the request. Therefore, the result of this first operation is two distinct clouds:

1. one containing all the points not selected by RANSAC;
2. the other represents the selected plane, which is then removed from the classification phase.

Since, as previously described, every piece of the cloud is analyzed separately; the centroid of the point cloud called "2" in Figure 3 (bottom right of the figure) is used as the relative center of the cloud called "1" (upper right of the figure). The original coordinates of the relative center are saved in the dataset and then the centroid is reset to the origin (0,0,0). This process allows tunnels to be analyzed regardless of elevation or geographical position.
The data from point cloud 1 are further segmented using the DBSCAN algorithm. Precisely, the DBSCAN is trained on both the coordinates of the points and their normal vectors, meaning that the clusters account for both location and direction. The DBSCAN uses two parameters to define clusters of points with homogeneous density. The first is '*epsilon*', defining the amplitude of a hypothetical radius drawn around the point. The second is *'minimum samples'*, specifying the number of points that must fall within the epsilon neighborhood of the core point and therefore included in the cluster. Points that do not fulfill this condition are considered noise points and hence excluded from the procedure.
It is necessary to evaluate if the DBSCAN algorithm is sufficiently suitable for segmenting the point clouds into subcomponents to be later classified. To measure the effectiveness and reliability of the process, we need to show that the DBSCAN is correctly separating the different components of the tunnel. Specifically, we want to ensure that the DBSCAN is not clustering together two or more different elements of the tunnel. Remark that the usual clustering evaluation metrics are somehow meaningless in this scenario, as any clustering partition that does not mix up different tunnel parts is equally preferable. Hence, we consider a segment of each of the four tunnels and create an oracle that labels all the points in the cloud correctly. We then run DBSCAN under different hyperparameter values that progressively increase the number of clusters. All the points in each cluster are

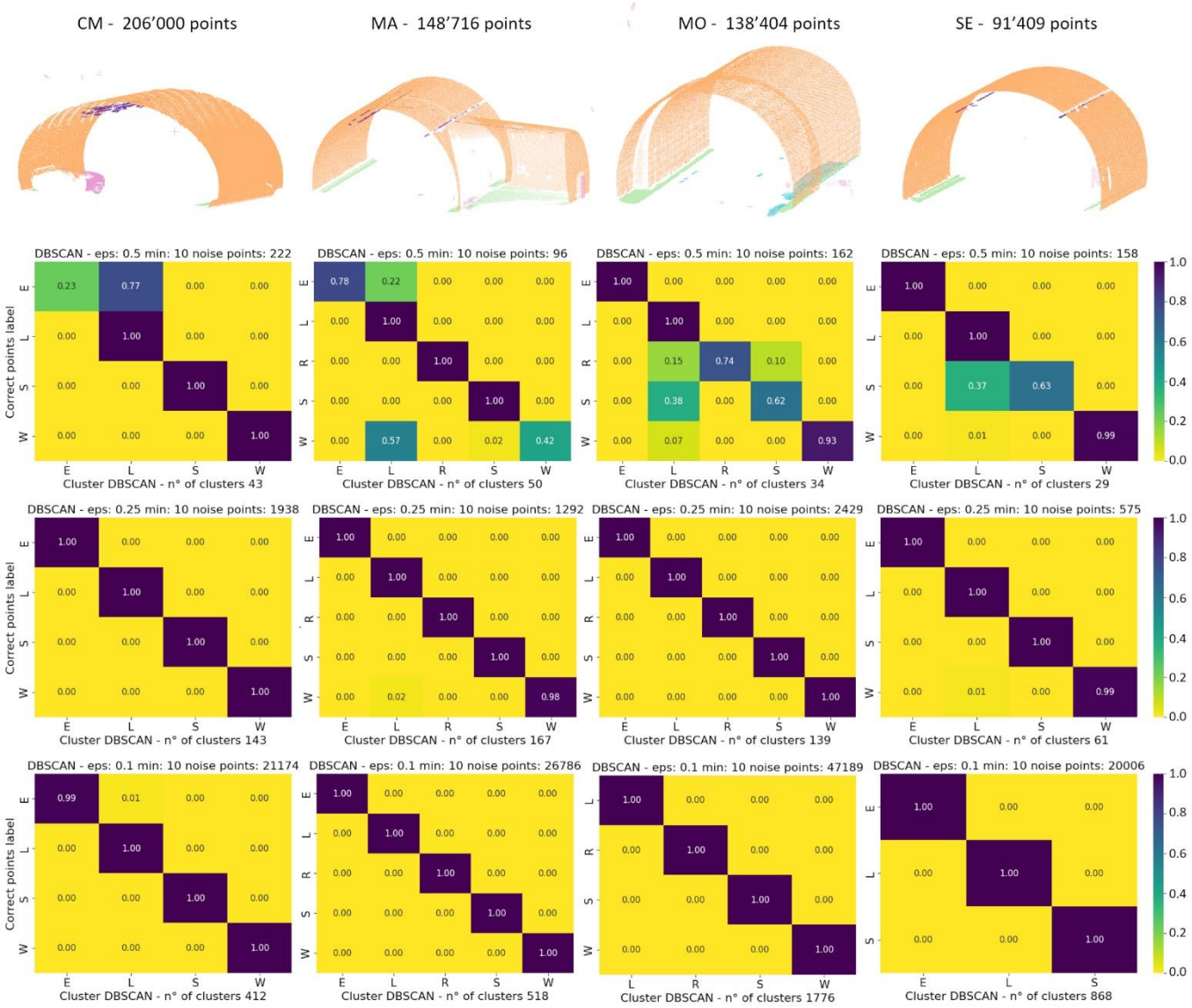

Figure 4. Confusion Matrixes of DBSCAN process. On the columns segments of different tunnels and on the rows the results produced by different combinations of epsilon and minimum samples.

subsequently labeled according to the class with the highest frequency from the oracle. In short, if the DBSCAN is creating clusters with points belonging to a single class, the aforementioned procedure would recover the oracle. Figure 4 shows the confusion matrices resulting from comparing the oracle with the labels that are produced with the DBSCAN on different tunnels (columns) and under different hyperparameter values (rows). As we can see from the figure, as the number of clusters increases, the algorithm segments the cloud more and more precisely, but the number of points that are classified as noise also increases, lowering the precision of the final cloud. For example, in the bottom-right matrix, the DBSCAN could no longer cluster the points from the WASTE class as they were too sparse. This suggests that a higher number of clusters is preferred, but overly finer partitions should be avoided. In particular, a segmentation that produces between 100 and 200 clusters is often sufficient to divide the elements correctly. However, as with RANSAC, we leave the final say on the choice of hyperparameters to technicians.

In our application, we varied the hyperparameters tunnel by tunnel. Specifically, for CM we use 0.22 as eps and 11 as min, for MA 0.24 as eps and 10 as min, for MO 0.18 as eps and 8 as min and SE 0.20 as eps and 9 as min.

Figure 5 illustrates the algorithm's execution on a CM tunnel segment which, as shown in Figure 3, is the densest cloud among those analyzed. For this analysis, epsilon was set to 0.25 and minimum

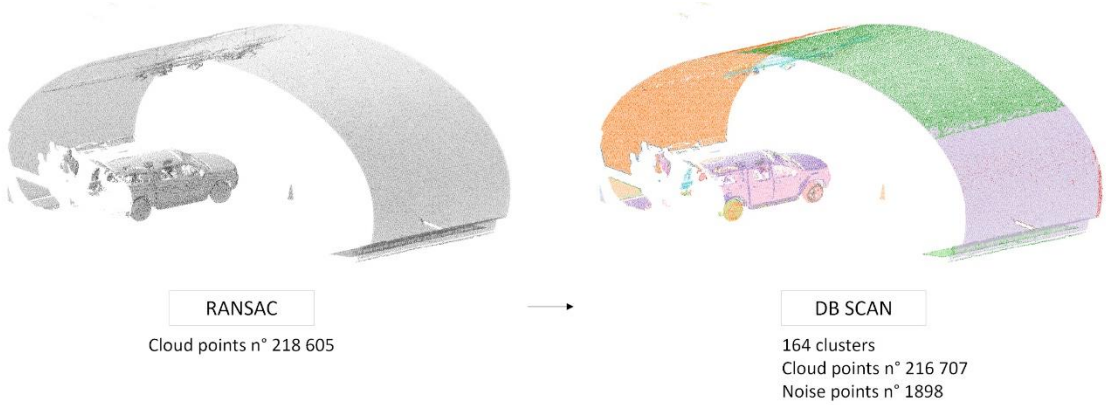

Figure 5. From left to right input and output of the DBSCAN algorithm. Different colors represent different clusters.

samples to 13. Using these parameters 164 clusters were produced, excluding noise points and clusters with less than 15 points, as we consider them too small to be significant. Each cluster identified by the DBSCAN was then associated with some covariates computed during the initial phase of the study. These covariates are going to be basic summary statistics of the cluster, which characterize the cluster properties and inform the classification algorithm.

Precisely, following the names in Table 2, the resulting covariates to be used for classification are:

- CentroidX, CentroidY, CentroidZ: the average of the coordinates of each point making up the cluster, the relative center is calculated during RANSAC;
- NormCentroidX, NormCentroidy, NormCentroidZ: the average of the normal vectors of the points constituting the cluster and representing its orientation;
- NumPoints: number of points in the cluster;
- Volume: the space the cluster occupies if enclosed in a bounding box;
- Density: the volume to several points ratio.

Each cluster is uniquely identified with a code organized as follows [Gallery code]_[Segment number]_[Cluster number]. Table 2 shows an example of our processed data, afterwards each row will be associated with a class.

Table 2 Some of the processed data from CM tunnel.

ID	CentroidX	CentroidY	CentroidZ	NormCentroidX	NormCentroidY	NormCentroidZ	NumPoints	Volume	Density
CM_28_0	0,174	3,576	2,641	-0,594	-0,683	-0,198	91322	388,277	0,004
CM_28_1	0,147	2,894	-0,021	-0,027	-0,010	0,995	4818	1,968	0,000
CM_28_3	-0,824	-6,418	5,132	-0,471	-0,524	0,674	12184	45,170	0,004
CM_28_5	-1,576	-3,638	2,029	0,236	-0,254	-0,932	68	0,043	0,001
CM_28_6	-1,333	-3,857	5,648	0,208	0,225	-0,909	222	0,250	0,001
…	…	…	…	…	…	…	…	…	…
CM_28_151	0,713	-2,109	1,331	-0,083	-0,099	-0,975	23	0,001	0,001
CM_28_153	0,842	-2,145	0,897	0,791	-0,310	-0,015	638	0,080	0,002
CM_28_158	0,794	-2,138	1,315	0,369	-0,075	0,810	87	0,003	0,001
CM_28_161	1,447	-3,911	4,706	0,083	0,129	-0,938	402	0,284	0,001
CM_28_162	1,075	-8,398	5,139	-0,502	-0,551	0,662	945	0,720	0,001
CM_28_163	0,871	-3,321	1,072	-0,507	-0,732	-0,309	53	0,007	0,001

4.3 Classification

We now delve into the task of classification, which will allow us to automatically detect (after training) parts of the point cloud to be maintained. Since this problem requires a supervised learning

approach, we need to manually label the resulting clusters from the previous section to properly train classification algorithms and so automate the process, with little or even absent human interaction. Note that, differently from vanilla segmentation problems [28] we are labeling the clusters and not the points, an extensive discussion on the advantages of this tabular data approach compared to segmentation methods can be found at the end of Section 2.

Five different classes are designed:

- Lining: points in the cloud belonging to the tunnel structure and cover;
- Equipment: points related to the electrical, piping, and ventilation system;
- Sidewalk: points describing the raised side platforms of the tunnel;
- Road: road points that are not clustered by RANSAC due to defects in the surface or non-planar road geometry;
- Waste: points that do not belong to any other classes.

The dataset is constructed through an iterative data visualization process where the outputs of the DBSCAN are manually labeled by assigning a class to each cluster, see Figure 6. Table 3 reports the breakdown of the labeled dataset. *A* total of 9203 clusters were classified resulting in: 27% being Lining, 27% being Equipment, 21% being sidewalks, 4% being road and 16% being Waste.

The dataset is then divided into two parts, 71% of the data is used to create a training dataset while the remaining 29% is used to test the algorithm and evaluate its accuracy. For the training and test split we adopt a somewhat uncommon approach in machine learning. We selected a contiguous portion in each of the four tunnels, and reserved it for the test set, using the remaining data for training. This choice reflects a realistic workflow as our test set resembles as much as possible a

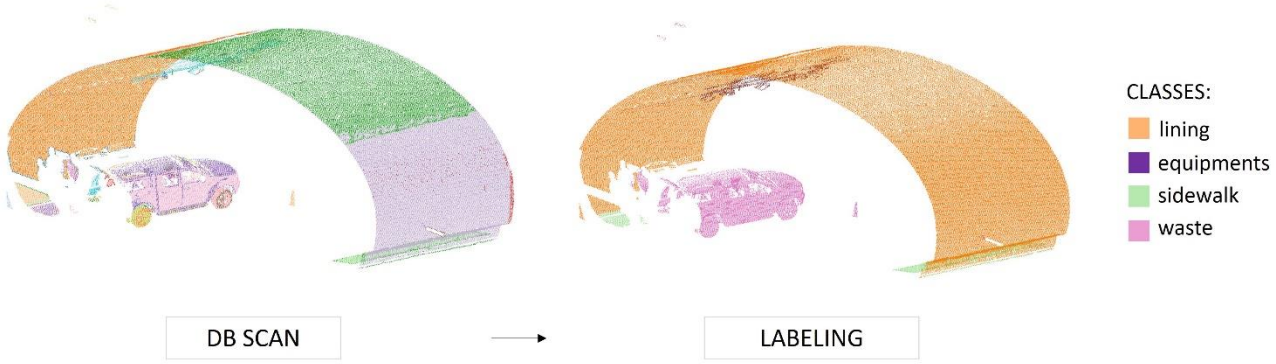

Figure 6 From left to right input and output of the labeling process after DBSCAN.

real tunnel.

A blind independent sampling for the training and test split would produce clusters that are not necessarily contiguous, and the resulting test performance would be unreliable. Note that this approach is not only measuring the test performance of the classification algorithm but also its robustness in new tunnel classification. Indeed, the least drop in accuracy at test time highlights the most robust approach when new data arrives.

Table 3 Dataset summary across classes.

	LINING	EQUIPMENT	SIDEWALK	ROAD	WASTE	Total	
CM	952	740	209	1	693	2596	28%
MA	905	1386	757	97	286	3431	37%
MO	361	366	602	277	455	2061	22%
SE	266	460	344	13	32	1115	12%
Total	2484	2952	1912	388	1466	9203	
	27%	32%	21%	4%	16%		

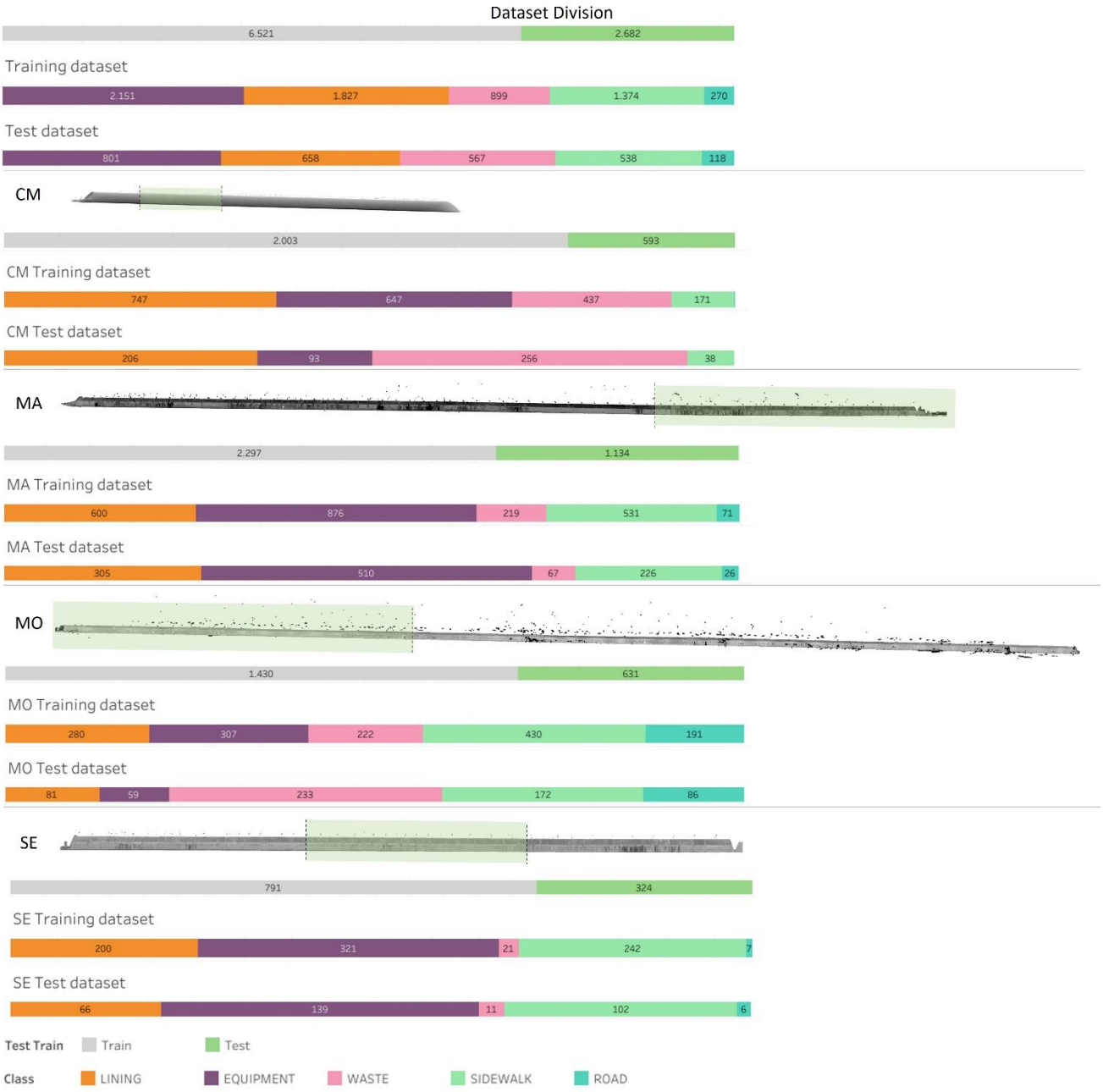

Figure 7. Division of the tunnels into training and test sets. Test sets are underline in green.

Figure 7 shows the aforementioned split with the test set highlighted in green. In particular, the CM tunnel is split to include in the test set a part of the tunnel with several waste elements, the MA and MO were separated to include the tunnel entrances and the test portion of MO contains a narrowing section, finally a central piece of SE was taken to add the standard gallery conformation to the test set.

Random forest [29], k-nearest neighbor [30], and naïve Bayes [31] are just some examples of the various classification algorithms to be considered. The choice of the classification algorithm is application dependent and ideally, we would like to try as many as possible to ensure that we are selecting the one that best suits our framework. To ensure enough algorithms are included in our model selection phase we use the Python library “lazypredict”, which allowed us to perform a test run on more than 20 different classification algorithms. Specifically, we divide our training set in training 1 and validation 1, following standard independent splitting (70% training 30% validation), and then divide our training 1 again in training 2 and validation 2 (70% training 30% validation), as

training 2 and validation 2 will be used for model selection and training 1 and validation 1 will be used for hyperparameters tuning. Within lazypredict, we then train our classifiers on training 2 compute accuracy on validation 2, and report the results in Table 4. Interestingly the top four algorithms are all ensemble methods based on decision trees, i.e. XGBoost [32], Extra Tree [33], LightGBM [34], and Random Forest [29], suggesting that tree-based models are more suitable to our classification problem. As an additional step, given the popularity of deep learning, we also fit fully connected neural networks [35]. We run experiments on different architectures (from 2 to 5 layers with either 1028, 2056, 4096, or 8192 hidden units) and under different activation functions (either "relu" or "tanh").

We found the accuracy on validation 2 to be the best when considering two layers of 4096 hidden units and a "relu" activation function. As the accuracy on validation 1 of the neural network was smaller than the tree-based algorithms we decided to exclude it from the rest of the study. This is not surprising as neural networks are known to be outperformed in the context of tabular data [36], and even more complicated deep learning approaches, like convolutional neural networks [37] and transformers [38], are not guaranteed to be better. Whether we have found the best classification algorithms or not, the paper aims to provide a clear pipeline on how to automate the process of cleaning point cloud datasets and we leave further analysis to future works.

From our model selection phase, we consider the top four algorithms and perform fine tuning of the hyperparameters by using training 1 and validation 1. Indeed, lazypredict performs blind training without tuning the hyperparameters of the methods, which can significantly boost performance [39]. Note that validation 1 is not used when performing model selection with lazypredict guaranteeing an unbiased estimate of the accuracy during the tuning phase. Each algorithm has a wide collection of hyperparameters to tune. We decided to focus on a maximum of 4 to 5 hyperparameters, whose choice was based on the documentation of the considered algorithms. We then set a grid of popular values per hyperparameter, train using training 1, and measure accuracy using validation 1. The output of the tuning step is then a 4 to 5 dimensional array of accuracy from which we can select the "best" combination of hyperparameters.

Table 4 Output from lazypredict reporting the considered algorithms in terms of accuracy

Model	Accuracy	Balanced Accuracy	F1-Score	Time (sec.)
XGBClassifier	*0,9272*	*0,8992*	*0,9263*	*2,4524*
LGBMClassifier	*0,9243*	*0,8963*	*0,9232*	*0,2385*
ExtraTreesClassifier	*0,9287*	*0,8962*	*0,9276*	*0,1786*
RandomForestClassifier	*0,9184*	*0,8901*	*0,9171*	*0,5847*
BaggingClassifier	*0,8971*	*0,8701*	*0,8954*	*0,1499*
LabelPropagation	*0,8941*	*0,8518*	*0,8936*	*0,2993*
LabelSpreading	*0,8941*	*0,8518*	*0,8936*	*0,4061*
DecisionTreeClassifier	*0,8735*	*0,8306*	*0,8733*	*0,0267*
ExtraTreeClassifier	*0,8463*	*0,8228*	*0,8467*	*0,0065*
KNeighborsClassifier	*0,8647*	*0,8202*	*0,8628*	*0,0401*
QuadraticDiscriminantAnalysis	*0,6934*	*0,7415*	*0,6516*	*0,0160*
GaussianNB	*0,6676*	*0,7320*	*0,6224*	*0,0078*
SVC	*0,8213*	*0,7319*	*0,8161*	*0,1328*
NearestCentroid	*0,5485*	*0,5703*	*0,5386*	*0,0686*
LogisticRegression	*0,7044*	*0,5655*	*0,6622*	*0,0253*
SGDClassifier	*0,6743*	*0,5620*	*0,6281*	*0,0724*
BernoulliNB	*0,6456*	*0,5488*	*0,6060*	*0,0117*
PassiveAggressiveClassifier	*0,6191*	*0,5420*	*0,5783*	*0,0395*

AdaBoostClassifier	*0,4684*	*0,5298*	*0,3449*	*0,2043*
CalibratedClassifierCV	*0,6765*	*0,5279*	*0,6220*	*0,0684*
LinearDiscriminantAnalysis	*0,6625*	*0,5088*	*0,5994*	*0,0582*
LinearSVC	*0,6566*	*0,5084*	*0,5868*	*0,2495*
Perceptron	*0,5610*	*0,4565*	*0,5490*	*0,0166*
RidgeClassifier	*0,6051*	*0,4341*	*0,4967*	*0,0115*
RidgeClassifierCV	*0,6051*	*0,4341*	*0,4967*	*0,0195*
DummyClassifier	*0,3500*	*0,2000*	*0,1815*	*0,0059*

Figure 8 graphically shows how accuracy can vary depending on the choice of hyperparameters. To simplify visualization, we report the most influential parameters from left to right, where the more we move to the right the more parameters we fix to the optimal or suboptimal values, more details are available in the Github repository. We can now extract the best combination of hyperparameters and move the performance onto the test set, which will be discussed in Section 4.5.

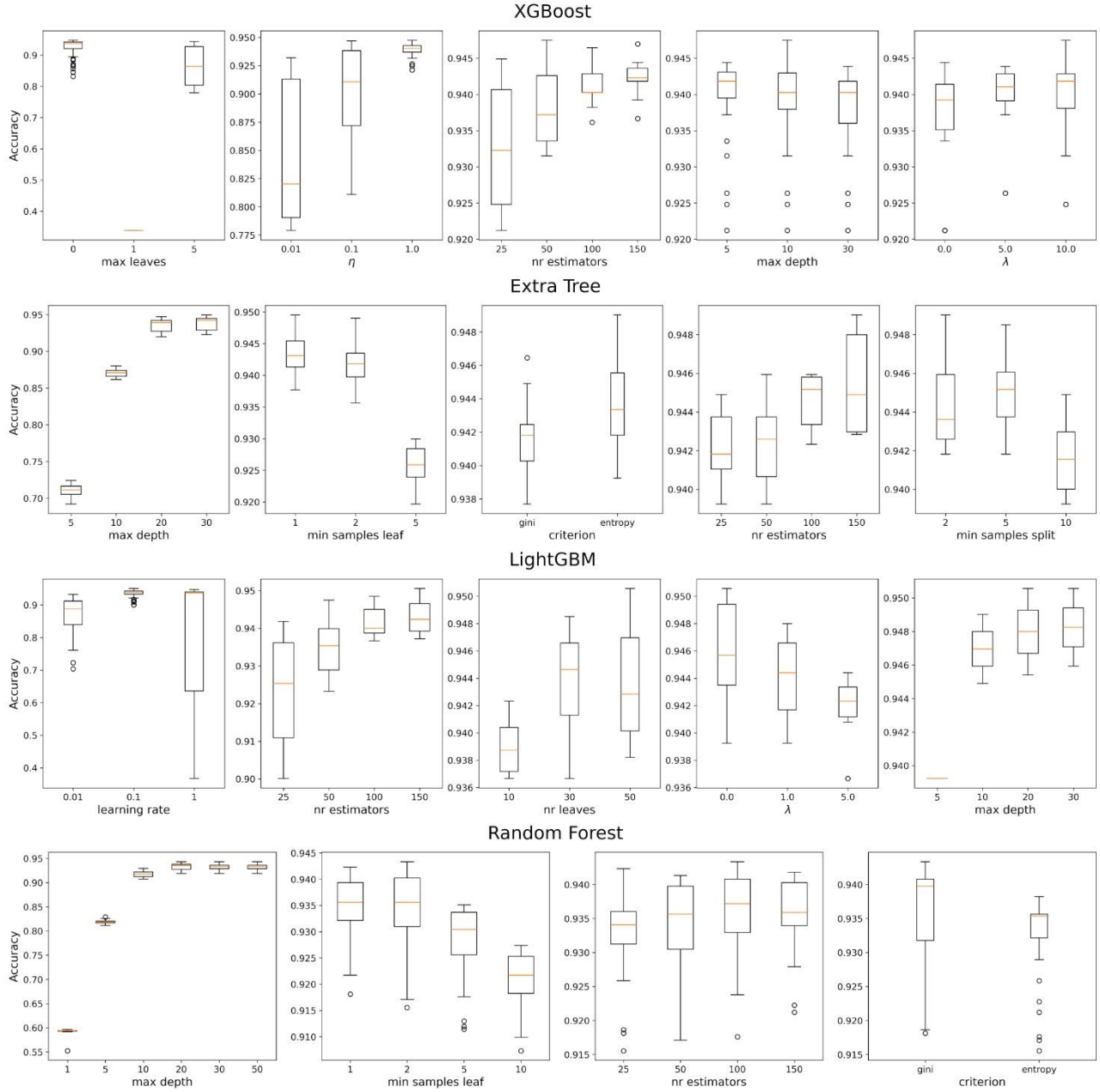

Figure 8. Box plots report the accuracy on validation 1 when varying hyperparameters. Their box plots are created by slicing the multidimensional array of accuracy, where some of the hyperparameters are also set to optimal values to improve visualization.

4.4 BIM model creation

Once the classification process is finished, the points describing the different classes analyzed can be extracted, at this point, it is necessary to discuss how this data can be used to reproduce the tunnel geometry inside a BIM model. Using the data linked to the L, S, and R classes, a point cloud containing points related to the tunnel structure, the road, and the side platforms is obtained. This can be used as a new basis for cross-section extraction, in this way the cleanup required to reconstruct the geometry from the extracted sections described in Section 2 is greatly reduced. Using this method of reconstruction, however, the final geometry will have no information regarding the morphology of the tunnel between the different sections. Depending on the proposed uses of the BIM model that information may be necessary as it ensures better correspondence with reality.
Figure 9 shows the proposed strategy; using geometric reconstruction algorithms, the point cloud is converted into a surface and then simplified to allow the creation of a BIM model. This provides a file that can be easily imported into design programs used by engineering firms.
Figure 9 shows images related to the creation of the BIM model containing the lining information of the SE Gallery, the same reconstruction process was also carried out on the other tunnels.

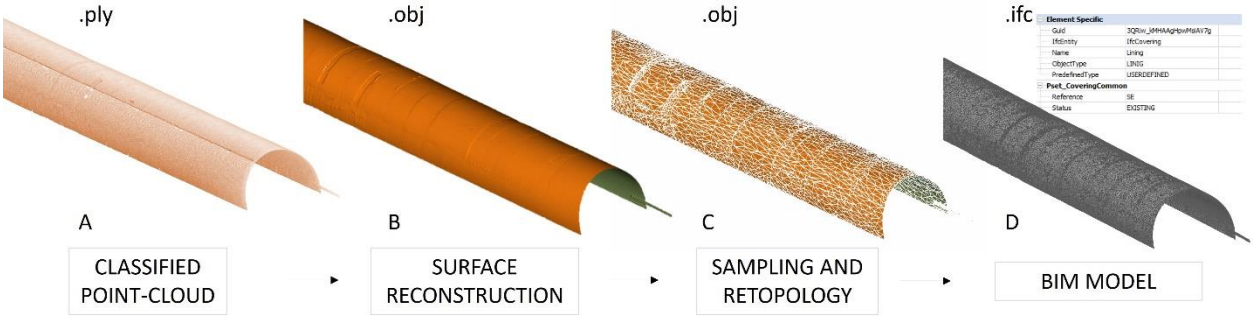

Figure 9 BIM model creation workflow. From left to right, classification of the point cloud, surface reconstruction via Poisson algorithm, geometry retopology and BIM model creation with geometric and alphanumeric information. On top of the images computer format and graphical representation.

The first step, A in the image, shows the selection of points that belong to class L, that portion consists of 2,594,770 points of the 4,025,112 that formed the original point cloud. To reconstruct the surface in its totality, it is necessary to use surface reconstruction techniques.
Several algorithms enable this operation, and they depend strongly on the quality of the survey. During experiments, good results were obtained through reconstruction based on Poisson's algorithm [40]. This method allows the surface reconstruction by triangulating the position of the points using their normal vectors. The result is a precise surface without holes on which the details of the surveyed tunnel can be appreciated. Depending on the density of the cloud, the Ball pivoting algorithm can also give good results. As described in [41] the algorithm recreates the surface using p-balls (a ball of radius p). Starting from each point a p-ball is placed and it will form a triangular face with the first two points intersecting it, resulting in a complete surface that closely reproduces the gallery. This method is computationally fast but high quality of starting data is essential to obtain a continuous surface without holes. Image B in Figure 8 shows the result of the reconstruction with the defects of the structure visible on the surface, the algorithm was run on a .ply format file containing data related to the position and normal vector of the points. The model created by this operation has 2’594’770 vertices, equal to the number of initial points, and 5’179’917 faces for a resulting .obj file size of 262’833kb.
The file size is not suitable for the creation of a BIM model because, to be used easily and with different tools, the geometries contained in the model must be optimized and the information needed to describe the project must be added to them. Surface simplification is done through a process of sampling and retopology in which the number of vertices and faces is decreased while keeping the resulting shape close to the original. A detailed analysis of methods for mesh simplification is done by [42]. Image C in Figure 8 shows the result of such an operation on the gallery by applying a criterion similar to the one described in [43] and setting a maximum error from the original geometry equal to 1cm, at the end of the operation the .obj file measures 12’697kb.
This optimized mesh can be used as the basis for the creation of the BIM model. The information

model is created by writing .ifc files according to the IFC4 standard [44]. To create the file correctly, the individual components must be assigned to the correct IFC class. In image D of Figure 8, the gallery structure was classified as IfcCovering with an ObjectType set to LINING. Other information within the PropertySets "Pset_CoveringCommon" is also added to the object, these report the gallery encoding "SE" and the coverage status "EXISTING". At the end of the process, the .ifc file measures 21,786kb, a dimension that allows its manipulation in BIM authoring software. At this point, the mesh derived from the point cloud can be used for the creation of a more detailed model within BIM authoring software. The most commonly used strategy is to create solid elements representing segments of the existing gallery. The inner surface of the solid elements is created with the point cloud, while survey data are used for the outer surface. These allow, for example, Boolean operations to precisely calculate the concrete volumes to be removed for the renovation of the structure.

A feature of the BIM methodology is the possibility of updating and adding information to the model to increase the details of the different elements and improve the communication of information during the design and maintenance phases. Thanks to the exposed methodology this capability of BIM models can be fully utilized, streamlining the processes of geometry restitution and optimization.

4.5 Results

This section discusses the results obtained during the point cloud classification in Section 4.3. The classification results of the test set are shown in Table 5.

Table 5 Test set accuracy of the four best algorithms. The results are divided by tunnel and class, whole dataset is reported at the bottom of the table.

			RandomForest		ExtraTree		XGBoost		LightGBM	
ID	Class	Total clusters	Correct	Accuracy	Correct	Accuracy	Correct	Accuracy	Correct	Accuracy
CM	Lining	206	184	89,32%	180	87,38%	183	88,83%	179	86,89%
	Equipment	93	87	93,55%	89	95,70%	88	94,62%	87	93,55%
	Sidewalk	38	35	92,11%	37	97,37%	34	89,47%	36	94,74%
	Road	0	0	0,00%	0	0,00%	0	0,00%	0	0,00%
	Waste	256	200	78,13%	193	75,39%	200	78,13%	197	76,95%
	Total	593	506	85,33%	499	84,15%	505	85,16%	499	84,15%
MA	Lining	312	294	94,23%	294	94,23%	299	95,83%	291	93,27%
	Equipment	390	339	86,92%	363	93,08%	325	83,33%	356	91,28%
	Sidewalk	265	232	87,55%	242	91,32%	216	81,51%	250	94,34%
	Road	48	36	75,00%	33	68,75%	33	68,75%	28	58,33%
	Waste	166	121	72,89%	128	77,11%	119	71,69%	120	72,29%
	Total	1181	1022	86,54%	1060	89,75%	992	84,00%	1045	88,48%
MO	Lining	81	74	91,36%	68	83,95%	75	92,59%	67	82,72%
	Equipment	59	57	96,61%	59	100,00%	59	100,00%	59	100,00%
	Sidewalk	172	171	99,42%	171	99,42%	170	98,84%	172	100,00%
	Road	86	68	79,07%	69	80,23%	68	79,07%	61	70,93%
	Waste	233	167	71,67%	152	65,24%	169	72,53%	155	66,52%
	Total	631	537	85,10%	519	82,25%	541	85,74%	514	81,46%
SE	Lining	81	79	97,53%	81	100,00%	79	97,53%	79	97,53%
	Equipment	138	136	98,55%	136	98,55%	136	98,55%	137	99,28%
	Sidewalk	101	99	98,02%	101	100,00%	101	100,00%	101	100,00%
	Road	4	2	50,00%	3	75,00%	3	75,00%	3	75,00%
	Waste	1	1	100,00%	1	100,00%	1	100,00%	1	100,00%

	Total	325	317	97,54%	322	99,08%	320	98,46%	321	98,77%
	Lining	680	631	92,79%	623	91,62%	636	93,53%	616	90,59%
	Equipment	680	619	91,03%	647	95,15%	608	89,41%	639	93,97%
	Sidewalk	576	537	93,23%	551	95,66%	521	90,45%	559	97,05%
	Road	138	106	76,81%	105	76,09%	104	75,36%	92	66,67%
	Waste	656	489	74,54%	474	72,26%	489	74,54%	473	72,10%
	Total	2730	2382	87,25%	2400	87,91%	2358	86,37%	2379	87,14%

As shown in Table 5, the classification accuracy on the test set depends on the considered tunnel, indicating that different morphologies of the tunnels may affect classification. The table presents the classification results divided by gallery, on the rows, and classification algorithm, on the columns.

Table 6 Alternative metrics to accuracy calculated for the four best algorithms.

Algorithm	Accuracy clusters	Balanced accuracy clusters	Weighted F1-score clusters	Accuracy points	IoU points	mIoU points
XGBoost	0.87253	0.85681	0.87122	0.99291	0.98591	0.84299
ExtraTree	0.87912	0.86153	0.87661	0.99138	0.98291	0.79297
LightGBM	0.86374	0.84660	0.86295	0.99280	0.98570	0.85388
RandomForest	0.87143	0.84076	0.86900	0.99215	0.98443	0.82116

Each row shows the analyzed gallery and reference class with the total number of clusters, while columns show the results in terms of clusters correctly classified by the algorithm and their percentages. The last rows of the table present the aggregated results over the entire test set.
It can be observed that the algorithms have greater difficulty identifying the clusters corresponding to "Road" and "Waste," which are, on average, correctly classified 70% of the time. For the "Road" clusters, the inaccurate results are likely due to an insufficient number of clusters used to train the algorithm, as most of the points representing the road are identified during the first clustering step, see section 4.2. In contrast, for the "Waste" class, the results likely stem from the heterogeneity of the clusters within this class, making them more challenging to identify. Nonetheless, the final results are encouraging, considering the potential for improving the algorithm by adding new galleries to the training dataset. To enhance the robustness of our algorithms, several metrics were employed beyond mere accuracy on tabular data. Alongside the metrics presented in Table 4 and Table 5, the intersection over union (IoU) and the mean intersection over union (mIoU) are reported in Table 6, as they are widely used in semantic segmentation [45]. The IoU measures the overlap between the predicted mask and the labeled point cloud, while mIoU is the arithmetic means of the IoU values across all categories. In Table 6 the overall accuracy on the point cloud is also reported, note that this metric is highly favoring big clusters.
The bubble charts in Figure 10 show the distribution of errors made by the algorithms by featuring bubbles based on the number of points contained in each cluster, which is a graphical representation of the overall accuracy in Table 6. The first five columns of the image show the results divided by class, the last four columns are inherent to the galleries, indicated in grey. Looking at the last four columns, it can be easily observed that the errors are concentrated in the center of the graph, denoting that the algorithm struggles to recognize clusters with small number of points. It is also evident from Figure 10 that the Waste and Road classes contain the most errors and that these are independent of the cluster's points.
From the results presented in this section, it can be said that the developed algorithms are reliable, especially for the detection of points belonging to the tunnel lining, as out of a total of 5671081 points belonging to the structure only 0.08% are not part of the correctly classified clusters.

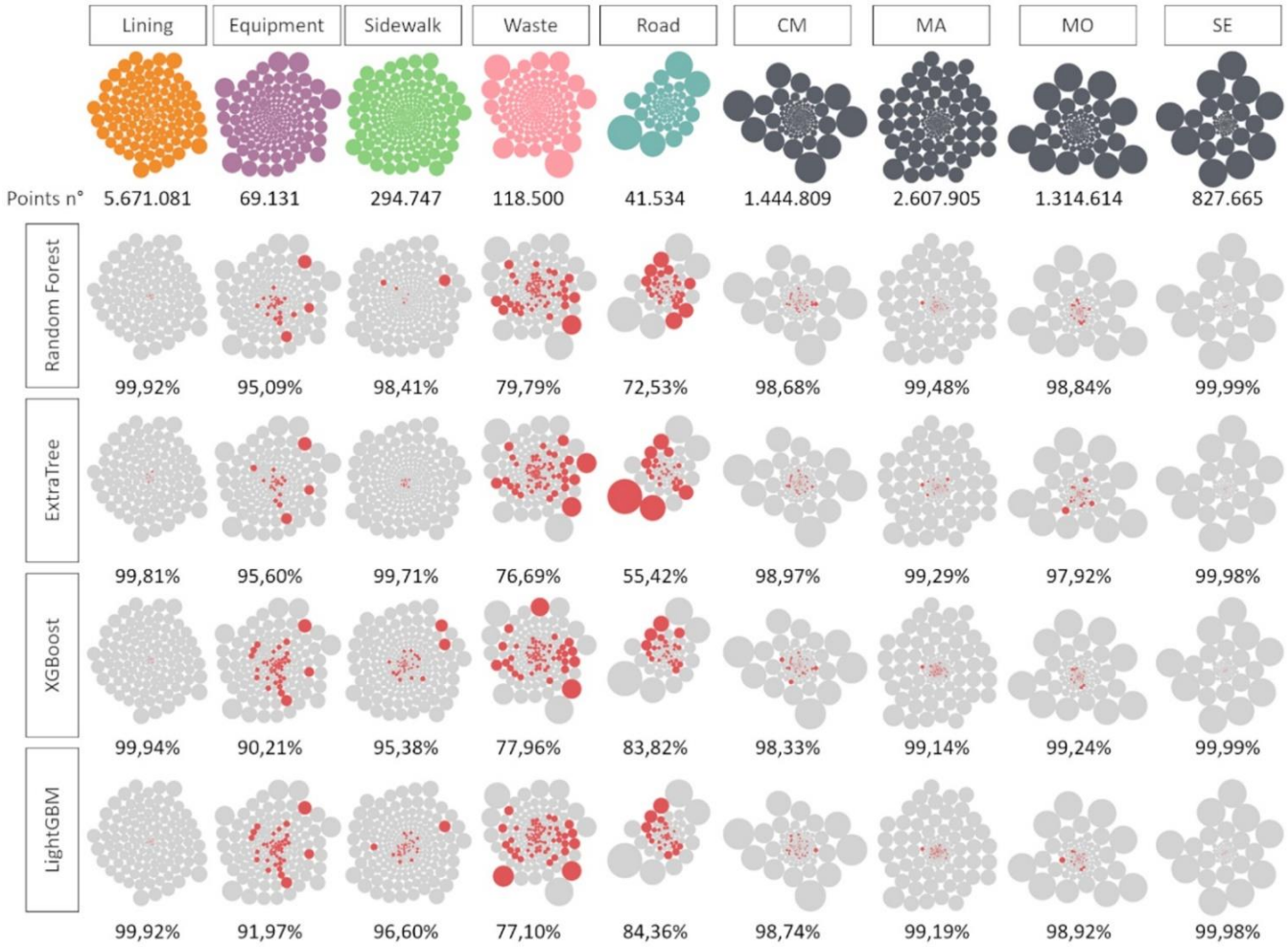

Figure 10 Bubble charts representing errors in test set. The size of the bubbles varies according to the number of points in the cluster. Columns from "Lining" to "Road" refer to the different classes, while columns from "CM" to "SE" consider the tunnels.

5. Conclusions

The paper presented a methodology for developing BIM models from laser scanner surveys using machine learning algorithms. Surveying using TLS is certainly an efficient way to describe the geometry of a tunnel, but its execution needs standards to obtain more reliable and clear data that can be used in a subsequent maintenance phase. Clustering and classification algorithms can enable better use of this data, providing more speed in analysis and information processing. The proposed approach can be scaled to different types of tunnels, and data from future classifications by technicians will provide new inputs for training the algorithms and improving their performances. Compared with traditional methodologies, the strategy presented in this contribution enables the possibility to quickly process large amounts of data and create BIM models that can be used for design and maintenance analysis. Even though the creation of a simple IFC model containing meshes is certainly not sufficient to design tunnel renovation work, the presented IFC model approximates the inner surface of the tunnel with a significant level of detail. Surely, this model must be implemented with historical data and surveys to reconstruct a parametric model satisfying the needs of engineering companies.

The results shown are satisfactory, although it should be noted that for the accurate reconstruction of the tunnel, in all its parts, manual intervention is still needed to correct wrong outputs and avoid impurities in the geometries resulting from the reconstruction process. For this reason, work is currently taking place on the creation of a graphical interface that allows better interaction with the classification results to speed up the correction process, make it user-friendly and enable the tool to be used in engineering studies.

The technology transfer process is a key element in our research, and we are currently engaging in such activities to enable the AECO supply chain to incorporate the proposed methodology into their workflows, and so reduce the time taken to analyze and clean up point clouds. The libraries and algorithms chosen are consistent with this aim, as they allow fast and accurate analysis without necessarily having to use powerful machines or remote clusters that can be costly for an engineering company.

Acknowledgments

Thanks to the support that was given by the engineering company TECNE this research could be carried out. In addition to funding the research, the company provided the tunnel data needed to be able to train and evaluate the algorithm. The continued support that has been provided by the company has also enabled the realization of work that is expected to have real application in the field. The authors also thank the editor and two anonymous reviewers for the invaluable feedback and for significantly improving the quality of the manuscript.

Funding: This work was supported by the engineering company TECNE. The authors acknowledge the company for providing the necessary data to conduct the analysis

References

[1] Senato della repubblica Italiana, Relazione concernente lo stato di attuazione degli interventi relativi all'adeguamento delle gallerie stradali della rete transeuropea, (2023), https://www.senato.it/service/PDF/PDFServer/DF/426541.pdf (accessed October 24, 2024).

[2] F. Di Stefano, S. Chiappini, A. Gorreja, M. Balestra, R. Pierdicca, Mobile 3D scan LiDAR: a literature review, Geomatics, Natural Hazards and Risk, 12 (2021) pp.2387–2429, https://doi.org/10.1080/19475705.2021.1964617.

[3] M.Q. Huang, J. Ninić, Q.B. Zhang, BIM, machine learning and computer vision techniques in underground construction: Current status and future perspectives, Tunnelling and Underground Space Technology, 108 (2021) pp.103677, https://doi.org/https://doi.org/10.1016/j.tust.2020.103677.

[4] C. Superiore, L. Pubblici, Linee guida per la classificazione e gestione del rischio, la valutazione della sicurezza ed il monitoraggio delle gallerie esistenti, Roma, (2022), https://cslp.mit.gov.it/sites/default/files/ALL%20DM%20247%20Allegato_1_-_LL_GG_gallerie_con_allegati_A-B-C1-C2-D.pdf (accessed October 24, 2024).

[5] BuildingSMART, OpenBIM, (2024), https://www.buildingsmart.org/about/openbim/ (accessed October 24, 2024).

[6] Q. Lu, X. Xie, A.K. Parlikad, J.M. Schooling, E. Konstantinou, Moving from building information models to digital twins for operation and maintenance, Proceedings of the Institution of Civil Engineers - Smart Infrastructure and Construction, 174 (2021) pp.46–56, https://doi.org/10.1680/jsmic.19.00011.

[7] M. Grieves, Digital Twin: Manufacturing Excellence through Virtual Factory Replication, (2014), https://www.3ds.com/fileadmin/PRODUCTS-SERVICES/DELMIA/PDF/Whitepaper/DELMIA-APRISO-Digital-Twin-Whitepaper.pdf (accessed October 24, 2024).

[8] BuildingSMART alliance, National Building Information Modeling Standard, National Institute of BUILDING SCIENCES, (2007), https://buildinginformationmanagement.wordpress.com/wp-content/uploads/2011/06/nbimsv1_p1.pdf (accessed October 24, 2024).

[9] L. Rimella, N. Rimella, GitHub Repository, (2024), https://github.com/LorenzoRimella/ML_tunneling.git (accessed October 26, 2024).

[10] W. Wang, W. Zhao, L. Huang, V. Vimarlund, Z. Ei Wang, Applications of terrestrial laser scanning for tunnels a review, Journal of Traffic and Transportation Engineering, 1 (2014) pp.325-337, https://doi.org/10.1016/S2095-7564(15)30279-8.

[11] R. Van Gosliga, R. Lindenbergh, N. Pfeifer, Deformation analysis of a bored tunnel by means

of terrestrial laser scanning, in: Image Engineering and Vision Metrology, ISPRS, Dresden, (2006) pp.167–172, https://www.isprs.org/proceedings/xxxvi/part5/paper/LIND_629.pdf (accessed October 24, 2024).
[12] T. Nuttens, A. De Wulf, G. Deruyter, C. Stal, H. De Backer, K. Schotte, A. DE Wulf, H. DE Backer, Application of laser scanning for deformation measurements: a comparison between different types of scanning instruments., in: Knowing to Manage the Territory, Protect the Environment, Evaluate the Cultural Heritage, FIG Working Week, Roma, (2012), pp.1-13, https://www.fig.net/resources/proceedings/fig_proceedings/fig2012/papers/ts09d/TS09D_nuttens_dewulf_et_al_5988.pdf (accessed October 24, 2024).
[13] Y.J. Cheng, W. Qiu, J. Lei, Automatic extraction of tunnel lining cross-sections from terrestrial laser scanning point clouds, Sensors, 16 (2016) pp. 1-16, https://doi.org/10.3390/s16101648.
[14] Z. Kang, L. Zhang, L. Tuo, B. Wang, J. Chen, Continuous extraction of subway tunnel cross sections based on terrestrial point clouds, Remote Sensing, 6 (2013) pp.857–879, https://doi.org/10.3390/rs6010857.
[15] J.D. Foley, M.A. Fischler, R.C. Bolles, Graphics and Image Processing Random Sample Consensus: A Paradigm for Model Fitting with Apphcatlons to Image Analysis and Automated Cartography, Commun ACM 24 (1981) pp.381–395, https://doi.org/10.1145/358669.358692.
[16] Y. Li, J. Shi, Z. Xiao, A Multilevel Point Cloud Classification Method for Underground Tunnels Based on Three-Dimensional Moving LiDAR Measurements, Mobile Information Systems 2022 (2022) pp.1-22, https://doi.org/10.1155/2022/6181182.
[17] M. Ester, H.-P. Kriegel, J. Sander, X. Xu, A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise, in: KDD'96: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, AAAI, Munchen, (1996) pp.226–231, https://file.biolab.si/papers/1996-DBSCAN-KDD.pdf (accessed October 24, 2024).
[18] A. Singh, N. Thakur, A. Sharma, A review of supervised machine learning algorithms, 3rd International Conference on Computing for Sustainable Global Development (INDIACom), IEEE, NewDelhi, (2016) pp.1310–1315, https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=7724478 (accessed October 24, 2024).
[19] Z. Xu, Z. Xie, X. Wang, M. Niu, Automatic Classification and Coding of Prefabricated Components Using IFC and the Random Forest Algorithm, Buildings 12 (2022) pp.1-22, https://doi.org/10.3390/buildings12050688.
[20] M. Huymajer, G. Paskaleva, R. Wenighofer, C. Huemer, A. Mazak-Huemer, IFC concepts in the execution phase of conventional tunneling projects, Tunnelling and Underground Space Technology 143 (2024) pp.1-13, https://doi.org/10.1016/j.tust.2023.105368.
[21] J. García-León, P. Sánchez-Allegue, C. Peña-Velasco, L. Cipriani, F. Fantini, Interactive dissemination of the 3D model of a Baroque Altarpiece: A pipeline from digital survey to game engines, SCIRES-IT 8 (2018) pp.59–76, https://doi.org/10.2423/i22394303v8n2p59.
[22] Q. Hu, B. Yang, L. Xie, S. Rosa, Y. Guo, Z. Wang, N. Trigoni, A. Markham, RandLA-Net: Efficient Semantic Segmentation of Large-Scale Point Clouds, (2019) pp.1-16, http://arxiv.org/abs/1911.11236.
[23] C.R. Qi, L. Yi, H. Su, L.J. Guibas, PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, Computer Vision and Pattern Recognition (2017) pp.1-14, http://arxiv.org/abs/1706.02413.
[24] S. Fan, Q. Dong, F. Zhu, Y. Lv, P. Ye, F.Y. Wang, SCF-Net: Learning spatial contextual features for large-scale point cloud segmentation, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society, (2021) pp.14499–14508, https://doi.org/10.1109/CVPR46437.2021.01427.
[25] L. Zhan, W. Li, W. Min, FA-ResNet: Feature affine residual network for large-scale point cloud segmentation, International Journal of Applied Earth Observation and Geoinformation 118 (2023) pp.1-9, https://doi.org/10.1016/j.jag.2023.103259.
[26] K.P.. Murphy, Machine learning : a probabilistic perspective, MIT Press, (2012), https://github.com/kerasking/book-1/blob/master/ML%20Machine%20Learning-A%20Probabilistic%20Perspective.pdf (accessed October 24, 2024).
[27] A. Mohamed, A. Dhiya, M. Jamila, H. Abir, J. A. Ahmed, A Systematic Review on Supervised and Unsupervised Machine Learning Algorithms for Data Science, in Supervised and

Unsupervised Learning for Data Science, Springer Nature (2019) pp.3-22, https://doi.org/10.1007/978-3-030-22475-2.
[28] T. Hackel, N. Savinov, L. Ladicky, J.D. Wegner, K. Schindler, M. Pollefeys, Semantic3D.net: A new Large-scale Point Cloud Classification Benchmark, (2017) pp.1-9, http://arxiv.org/abs/1704.03847.
[29] L. Breiman, Random Forests, Mach Learn 45 (2001) pp.5–32, https://link.springer.com/content/pdf/10.1023/A:1010933404324.pdf (accessed October 24, 2024).
[30] T. Cover, P. Hart, Nearest neighbor pattern classification, IEEE Trans Inf Theory 13 (1967) pp.21–27, https://doi.org/10.1109/TIT.1967.1053964.
[31] P. Domingos, M. Pazzani, On the Optimality of the Simple Bayesian Classifier under Zero-One Loss, Mach Learn 29 (1997) pp.103–130, https://doi.org/10.1023/A:1007413511361.
[32] Distributed (Deep) Machine Learning Community, Scalable and Flexible Gradient Boosting, (2024), https://xgboost.ai/ (accessed October 24, 2024).
[33] scikit learn, ExtraTreesClassifier, (2024), https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.ExtraTreesClassifier.html (accessed October 24, 2024).
[34] Microsoft, Welcome to LightGBM's documentation!, (2024), https://lightgbm.readthedocs.io/en/stable/# (accessed October 24, 2024).
[35] I. Goodfellow, Y. Bengio, A. Courville, Deep Learning, MIT Press, (2016), https://www.deeplearningbook.org/ (accessed October 24, 2024).
[36] R. Shwartz-Ziv, A. Armon, Tabular data: Deep learning is not all you need, Information Fusion 81 (2022) pp.84–90, https://doi.org/10.48550/arXiv.2106.03253.
[37] K. O'Shea, R. Nash, An introduction to convolutional neural networks, Neural and Evolutionary Computing (2015) pp.1–11, https://doi.org/10.48550/arXiv.1511.08458.
[38] T. Lin, Y. Wang, X. Liu, X. Qiu, A survey of transformers, AI Open 3 (2022) pp.111–132, https://doi.org/https://doi.org/10.1016/j.aiopen.2022.10.001.
[39] P. Probst, M.N. Wright, A. Boulesteix, Hyperparameters and tuning strategies for random forest, Wiley Interdiscip Rev Data Min Knowl Discov 9 (2019) pp.1301–1320, https://doi.org/10.1002/widm.1301.
[40] M. Kazhdan, H. Hoppe, Screened poisson surface reconstruction, ACM Transactions on Graphics (ToG) 32 (2013) pp.1–13, https://doi.org/10.1145/2487228.2487237.
[41] F. Bernardini, J. Mittleman, H. Rushmeier, C. Silva, G. Taubin, The ball-pivoting algorithm for surface reconstruction, IEEE Trans Vis Comput Graph 5 (1999) pp.349–359, https://doi.org/10.1109/2945.817351.
[42] P. Cignoni, C. Montani, R. Scopigno, A comparison of mesh simplification algorithms, Comput Graph 22 (1998) pp.37–54, https://doi.org/10.1016/S0097-8493(97)00082-4.
[43] W.J. Schroeder, J.A. Zarge, W.E. Lorensen, Decimation of triangle meshes, in: Proceedings of the 19th Annual Conference on Computer Graphics and Interactive Techniques, Association for Computing Machinery, New York, NY, USA, 1992: pp.65–70, https://doi.org/10.1145/133994.134010.
[44] BuildingSMART, Industry Foundation Classes Version 4.2, (2024), https://standards.buildingsmart.org/IFC/DEV/IFC4_2/FINAL/HTML/ (accessed October 24, 2024).
[45] Z. Lu, T. Wu, Y. Dai, W. Li, Z. Su, Fine-grained Metrics for Point Cloud Semantic Segmentation, Computer Vision and Pattern Recognition (2024) pp.1-14, http://arxiv.org/abs/2407.21289.

Author Biography

Nicola Rimella is a Building engineer, who graduated at Politecnico di Torino in 2020. He has worked as a computational designer and R&D developer at a building digitization company. He is currently a PhD student in civil engineering at Politecnico di Torino, Italy. Contact him at nicola.rimella@polito.it.

Lorenzo Rimella is a postdoctoral researcher at the Universitá degli Studi di Torino, also affiliated with Collegio Carlo Alberto. His research on state-space models spans from epidemiological applications to skills ranking in competitive sport and traffic modeling. He completed his PhD in 2021 at the University of Bristol, studying how to scale up state-space models to high dimensions. Contact him at lorenzo.rimella@unito.it.

Anna Osello is a full professor at Politecnico di Torino, Italy, and she is in charge of the “Drawing to the Future” research group. Her research interests include augmented reality and building information modeling, and she has studied historical architectures and urban spaces. Osello received a PhD in drawing and survey of the building heritage at La Sapienza University of Rome. Contact her at anna.osello@polito.it.