Technical University of Munich

TUM School of Engineering and Design

Chair of Computational Modelling and Simulation

A Bottom-Up Approach for the Automatic Creation of a Digital Staircase Model Using Point Cloud Data and Parametric Prototype Models.

Masterthesis

for the Master of Science Program Resource-efficient and Sustainable Building

Author: Alexandra Bulla

[redacted] [redacted]

1. Supervisor:	Prof. Dr.-Ing. André Borrmann
2. Supervisor:	M.Sc. Mansour Mehranfar
Date of issue:	01. October 2023
Date of submission:	01. April 2024

Preface

In the master thesis, I am thankful for the persons who supported its realization. I convey my appreciation to M.Sc. Mansour Mehranfar for his expertise, guidance, feedback, support, and supervision.

Acknowledgment is also due to the School of Engineering and Design at the Technical University of Munich, particularly the Chair of Computational Modeling and Simulation, where the thesis was created and defended.

Lastly, I express gratitude to my parents and friends. Their encouragement has been a crucial foundation for the completion of the master's thesis.

Abstract

In the last decade, there has been a growing interest in the development of DT models for existing structures and infrastructures in the built environment. In this context, laser scanner technology emerges as a technology due to its ability to capture dense point clouds of the built environment with precise geometry and semantics. The creation of a digital model from PCD involves two main steps: firstly, semantic labeling of point cloud to separate objects, followed by the creation of the digital model. The conventional process of creating digital models is associated with challenges in processing point clouds and inferring the topological relation between elements. To address the challenges, the capabilities of Artificial Intelligence (AI) and Machine Learning (ML) capabilities in scene understanding are leveraged. The main objective is to develop an automated method for creating a digital model of staircase objects using PCD. To achieve this, a data-driven bottom-up approach is used to identify points corresponding to each instance of a staircase within the point cloud. Furthermore, an automated method is implemented to extract values of parameters required for creating the digital model. To create the digital model of the staircase, a library of parametric staircase models is collected. These models are defined by key parameters such as height, width, and length of each run within the staircase. Additionally, these models are made adaptable by defining the direction of the staircase and the number of landing treads. The extracted parameters from the point cloud data are used for the creation of the digital model based on the predefined staircase library. The result of testing the proposed approach on the dataset highlights its effectiveness with a mean error value of about 3.5cm in the estimation of the elements parameters.

Zusammenfassung

In den letzten Jahren ist das Interesse an der Entwicklung von DT Modellen für bestehende Strukturen und Infrastrukturen in der gebauten Umgebung gestiegen. In diesem Kontext erweist sich die Laserscanner Technologie als entscheidende Technologie aufgrund ihrer Fähigkeit, dichte Punktwolken der gebauten Umgebung mit präziser Geometrie und Semantik zu erfassen. Die Erstellung eines digitalen Modells aus der Punktwolke umfasst zwei Hauptschritte: erstens die semantische Beschriftung der Punktwolke zur Trennung von Objekten, gefolgt von der Unterscheidung der Objekte in der Punktwolke. Der herkömmliche Prozess der Erstellung digitaler Modelle ist mit Herausforderungen bei der Verarbeitung von Punktwolken und dem Ableiten der topologischen Beziehung zwischen den Objekten verbunden. Um dieser Herausforderung zu entsprechen, werden die Fähigkeiten künstlicher Intelligenz (KI) und maschinelles Lernen (ML) in der Szenenverarbeitung genutzt. Das Hauptziel besteht darin, eine automatisierte Methode zur Erstellung eines digitalen Modells von Treppenstrukturen unter der Verwendung dichter Punktwolken zu erstellen. Ein datengesteuerter Bottom-up-Ansatz wird verwendet, um Punkte zu identifizieren, die einer Treppe in der Punktwolke entsprechen. Darüber hinaus wird eine automatisierte Methode implementiert, um Werte für die erforderlichen Parameter des digitalen Modells zu extrahieren. Zur Erstellung des digitalen Modells der Treppenstrukturen wird eine Bibliothek parametrischer Treppenmodelle erstellt. Diese Modelle werden durch Schlüsselparameter wie Höhe, Breite und Länge jeder Stufe definiert. Zusätzlich werden diese Modelle angepasst, indem die Richtung der Treppe und die Anzahl der Treppenstufen definiert werden. Die extrahierten Parameter aus den Punktwolken werden für die Erstellung des digitalen Modells auf Grundlage der vordefinierten Treppenbibliothek verwendet. Das Ergebnis der Testung des vorgeschlagenen Ansatzes anhand des Datensatzes der Technischen Universität München zeigt dessen Effektivität mit einem mittleren Fehlerwert von etwa 3.5cm bei der Schätzung der Parameter.

Contents

List of Figures

List of Tables

List of Abbreviations

AI	Artifical Intelligence
AEC/FM	Architecture, Engineering and Construction/ Facility Management
ALS	Aerial Laser Scanner
BIM	Building Information Modeling
Brep	Boundary Representation
CSG	Constructive Solid Geometry
DBSCAN	Density-Based Spatial Clustering of Application with Noise
DT	Digital Twin
eps	Epsilon
FN	False Negative
FP	False Positive
IoU	Intersection over Union
IoT	Internet of Things
KI	Künstliche Intelligenz
LiDAR	Light Detection and Ranging
LOD	Level of Detail
MLS	Mobile Laser Scanning
ML	Machine Learning
minPts	Minimum Points
PCD	Point Cloud Data
RADAR	Radio Detection and Ranging
RANSAC	Random Sample Consensus
RGB	Red Green Blue
SONAR	Sound Navigation and Ranging
TLS	Terrestrial Laser Scanning
TN	True Negatives
TP	True Positives

1 Introduction

1.1 Motivation

The digital twin DT, initially arising in the manufacturing and automotive sectors, has found its way into Architecture Engineering and Construction (AEC) domain as a tool within the operation and maintenance phase. Ensuring the synchronization between the DT and its physical counterpart is a main challenge in the implementation of DT technology. The key lies in maintaining the system's capacity to update the digital model to mirror real-time changes in the physical entity. Therefore, monitoring, control and simulation will be necessary during the object's lifecycle, because its prediction of the future construction state allows to simulate and test preventive measures. However, the adoption of the DT concept in the construction industry remains in its beginning. The upward trend in research underscores a rising interest in exploring DT within the construction domain (Opoku et. al, 2021).

1.2 Objective

The main objective of the thesis is to propose an automated method for the creation of digital staircase models using PCD. An approach including the methodology, namely the bottom-up approach is utilized. To illustrate the proposed methodologies, a parametric model of a staircase is created, allowing for adjustments of various parametric values. In the final stage, the extracted parametric values within a geometric model of a staircase are verified by setting the created parametric model in comparison with the point cloud. Through this exploration, the effectiveness and accuracy of the proposed methodology in automating the recreation process is evaluated.

1.3 Structure

The thesis is organized as follows: In Section 2, a review of current literature associated with the topic related to the recreation of a digital model from PCD is represented, which serves as a foundation, providing background information for subsequent sections. Section 3 presents the developed methodology, which is described in detail from a theoretical standpoint. Section 4 showcases several case studies to demonstrate the feasibility of the proposed approach. Section 5 offers a discussion of the main findings of the study, along with future directions for research in the field. Finally, Section 6 provides a conclusion.

2 State of the Art

2.1 Digital Twinning of the Real World

Although various definitions of a DT in the built environment define each different components such as geometric or semantic information, objects or processes, the common essence in the definition of a DT lies in its ability to mirror a real-world asset in a virtual space, maintaining a continuous connection through real-time data exchange between the physical entity and its digital counterpart (Kritzinger et al., 2018).

In current academic discussions, there is an ongoing debate about whether the concept DT can be equated with the definition of a BIM. This debate arises from the parallel representation of an asset's functional and physical characteristics within both DT and BIM frameworks. While some scholars may find concordance in the assertion, dissenting viewpoints are also evident. Daskalova (2021) contributes to this discourse by providing a distinction between DT and BIM predicated on their purposes. A disparity articulated by Daskalova (2021) resides in the integration of real-time data within the DT. This difference serves as a factor for distinguishing DT from BIM. While BIM concentrates predominantly on the static representation of the designed and constructed state, it refrains from integrating real-time data (Daskalova, 2021).

Similarly, as stated by Pan (2023), a BIM primarily focuses on geometric and semantic information related to the design and construction phases of a facility. It functions as a digital representation of the physical and functional characteristics of the built environment's assets. On the other hand, a DT extends beyond the BIM concept by integrating real-time data from various sources, such as sensors and Internet of Things (IoT) devices throughout the entire lifecycle of a facility. This broader scope allows a DT to provide an up-to-date representation of the physical asset, enabling better monitoring, analysis, and decision-making across multiple sectors, including water or waste systems (Pan, 2023).

Lu and Brilakis (2019) introduce a conceptual framework that defines three forms of DT, namely the as-designed, as-built, and as-is DT. The as- designed DT is characterized by its utilization of data provided by designers prior to construction, serving as a tool during the design phase to simulate, visualize and optimize the planned design before its construction. In contrast, the as-built DT captures the physical realization of the constructed asset, emphasizing variations from the original design that may have occurred during the construction phase. This variant is crucial in capturing real-world conditions and deviations that occurred during and after the construction phase. The final variant of a DT is the as-is DT, which represents the state of the asset throughout its operational lifespan. This iteration of the DT is relevant during the asset's maintenance and operational phase, facilitating real-time monitoring, analysis, and management. The integration of real-time data provides insights into the current condition and performance of the asset (Lu & Brilakis, 2019). Lu posits that the central benefit of a DT resides in the automation of inspection processes. Through the incorporation of real-time data, the DT facilitates evaluations of the asset's condition, thereby effecting cost reductions in the oversight and maintenance of the built environment's assets (Lu & Brilakis, 2019).

2.2 Scan-to-BIM

A PCD captures precise and detailed as-built information, particularly valuable in the creation of as-built digital building model. The procedure involved in transforming PCD into an accurate representation of the existing structure or environment in the form of a BIM is commonly known as Scan-to-BIM. In other words, Scan-to-BIM is the process of utilizing PCD to reconstruct a BIM that reflects the real-world conditions of the built environment (Son et al. , 2015). At present, the manual processing of the scan-to-BIM procedure is both labor-intensive and error-prone. While LiDAR devices increase the scene capture process in surveys compared to traditional manual techniques, the time required for generating the corresponding BIM model tends to increase. The increased duration depends on factors such as the complexity and size of the structure, as well as the presence of building elements like windows, doors, and construction details (Rocha & Mateus, 2021). For instance, the time spent capturing through laser scanners is 2.82 hours, whereas the subsequent modeling demands 28 hours. Consequently, contemporary research focuses on automating the modeling process to reduce human-related labor and enhance efficiency (Pan, 2023).

As noted by Tang et al. (2010), the landscape of commercial software supporting the scan-to-BIM process is in the state of rapid progress. Despite the availability of various software solutions, no single program can comprehensively remodel an entire point cloud dataset. Consequently, there is a need to transfer information multiple times between different software tools to attain the desired outcome. This iterative process often results in the loss of information during data exchanges. Automating the process not only reduces labor costs but also accelerates project timelines by streamlining the generation of BIM models (Tang et al., 2010).

2.3 Point Cloud Data

The acquisition of environmental geometry is facilitated using laser scanning devices, leveraging the technology of LiDAR. This technology enables the generation of PCD, which serves as a detailed representation of the scanned environment. Gathering data using measuring tapes or conventional cameras is both time-consuming and prone to inaccuracies when implemented on a large scale. Consequently, laser scanning has emerged as the method for acquiring precise information about the built environment (Tang et al., 2010).

Depending on the intended application, distinct LiDAR devices, namely aerial laser scanners (ALS), terrestrial laser scanners (TLS), or mobile laser scanners (MLS), are employed. While these devices share the LiDAR technology, they diverge in their application. Aerial laser scanners ALS, for instance, are affixed to flying objects such as drones to capture terrain information from above. Mobile laser scanners MLS are mounted on platforms like cars and trucks, mapping urban environments. In contrast, terrestrial laser scanners TLS are mounted on fixed structures like tripods to capture detailed information surrounding the scanner's position (Olsen et al., 2010). In the field of AEC, TLS emerges as a method for surveying, monitoring, and inspecting the quality within the built environment. This technique is favored for its accuracy for capturing structural measurements, because they offer accurate and detailed as-built information (Abreu et al., 2023).

PCD is a collection of three-dimensional points that describe the surfaces of objects within a scanned environment. This dataset comprises XYZ coordinates, capturing the spatial positions of each point. Additionally, it includes information regarding RGB values, conveying color attributes, and surface normals, providing insight into the orientation of the points with references to the surfaces they represent (Bello et al., 2020).

Unlike, RADAR, which used radio waves and SONAR, which relies on sound waves, LiDAR employs light waves to capture information about the environment. The devices consist of an emitter, a rotator, and a photodetector. The emitter emits a pulse of laser light, traveling until it encounters an object. Upon reaching the object's surface, the laser beam reflects to the emitter in its original direction. To send beams in all directions, the emitter is placed on a continuously rotating device. The reflected light is then captured by the photodetector (Liu et al., 2021).

To calculate the distance between the device and the object, the LiDAR records the time the laser beam is emitted and the time until the reflected beam returns to the device. The difference between these times corresponds to the total travel time of the laser beam, which, due to reflection at the object's surface, is twice the distance traveled. Using the constant speed of light and the measured time, the distance is calculated with the equation:

$$distance = \frac{c}{2} \cdot \Delta t \qquad \textit{Eq.1}$$

Here c is the speed of light, Δt equals the time the light signal has travelled. Following this, trigonometric calculations, combined with the orientation information of the LiDAR sensor, transform the distance for each point into XYZ coordinates in the device's local coordinate system (Liu et al., 2021).

2.4 Point Cloud Semantic Enrichment

The semantic enrichment is described by the processing of PCD, particularly the assignment of a label to points using semantic segmentation and classification through traditional methods or AI techniques. The semantic segmentation of point clouds involves various approaches, primarily categorized as shape-based or region-based methods. These approaches, however, rely on predefined primitive shapes characterized by specific parameters and thresholds (Pan, 2023). To semantically segment a point cloud, machine learning algorithms are employed. Machine learning algorithms do not depend on parameters but are capable of learning existing patterns within a data structure. Machine learning algorithms are categorized into supervised and unsupervised machine learning algorithms. The subsequent section explores the segmentation of point cloud data, examining both traditional bottom-up and top-down approaches, characterized by shape or region-based methods, and machine learning algorithms. Within the realm of supervised machine learning, the section explores the use of classification trees as a tool for predictive modeling. Moreover, it provides insights into unsupervised machine learning, specifically emphasizing the clustering method known as Density-Based Spatial Clustering of Applications with Noise DBSCAN. The effectiveness of DBSCAN is clarified as a technique for grouping similar data points based on patterns, thereby contributing to the exploration of groups within the dataset.

2.4.1 Bottom-up and Top-down approach

According to Abreu et al. (2023), PCD analysis involves identifying features that represent characteristic geometric shapes. In the bottom-up approach, the process begins by selecting individual data points, collecting, and clustering them to form a geometric structure. For instance, Awwad et al., (2010) represented a workflow for detecting planar surfaces within unstructured point clouds using algorithms within the bottom-up approach (Awwad et al., 2010). On the other hand, the top-down approach reconstruction algorithms address uncertainty or missing data by incorporating knowledge about the structure's appearance and arrangement. This acquired knowledge is subsequently employed to synthesize information in regions where data is uncertain or absent (Abreu et al., 2023). Mehranfar et al., (2022) developed a workflow which uses networks to establish connections between spaces and objects (Mehranfar et al., 2022).

Region-based techniques leverage local features obtained from a surrounding neighborhood of each point to combine neighboring points sharing similar characteristics (Khaloo & Lattanzi, 2017). The process initiates by selecting a seed point, and a region is gradually expanded around this seed point by incorporating points with similar features. Figure 1 shows the algorithm applied to a point cloud to identify planar surfaces with similar z coordinates. The algorithm starts by choosing the point with the lowest z coordinate as the initial seed point. A specified threshold determines the acceptable difference in z coordinates for points to be included in the growing region. The algorithm iteratively grows the region until z coordinate differences exceed the defined threshold. After completing one iteration for a specific seed point, the algorithm selects a new seed point to initiate the new region growing process. This iterative procedure continues until the entire point cloud is segmented into regions with similar z coordinates (Khaloo & Lattanzi, 2017).

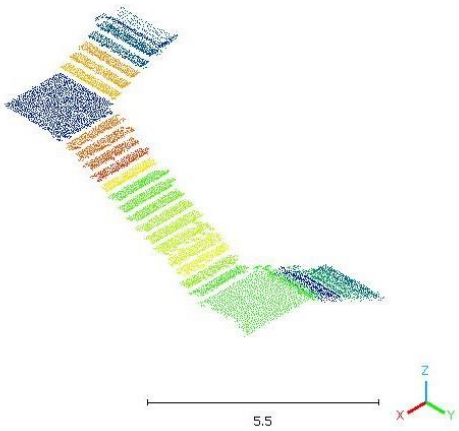


Figure 1: Result of the region growing algorithm for the separation of planar surfaces within the staircase.

Shape-based methods, such as the Random Sample Consensus RANSAC algorithm, are employed to identify groups of points conforming to specific primitive structures within a larger dataset. RANSAC detects parametrically defined primitives, including planes, cylinders, lines, ellipses, or circles, within PCD. For instance, when fitting a planar surface using RANSAC, the following parametric equation of a plane is utilized:

$$Ax + By + Cz + D = 0 \qquad \textit{Eq. 2}$$

Where A, B and C represent the normal vector of the plane, D is represented by a constant term on a plane and x y and z are coordinates of a point on the plane. The RANSAC algorithm starts with randomly selecting three points from the data and defining a planar surface equation based on the chosen points. By measuring the distance between the remaining points and the calculated plane, the algorithm aims to find the planar surface equation that results in most of the points lying in the same plane. According to Fayez et. al (2007) the algorithm requires specific input data, namely the corresponding point cloud and numeric parameters. These parameters serve two primary purposes: firstly, they determine the threshold for the number of points required to be considered part of the same plane, and secondly, they specify the maximum acceptable distance between a plane and the point to be categorized as an inlier. One of the algorithm's main function is to isolate points associated with walls, ceilings, or floors, as depicted in the accompanying Figure 2 (Fayez et al., 2007).

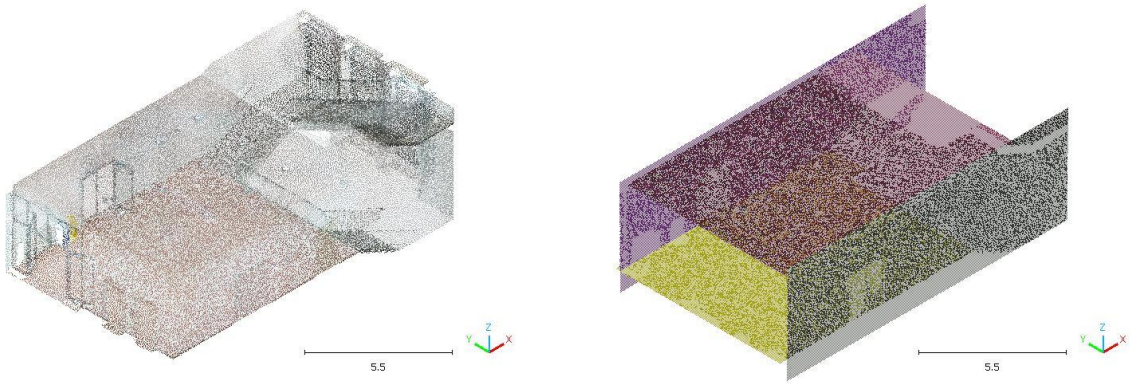


Figure 2: The result of the implementation of the RANSAC algorithm for the detection of planar surfaces.

2.4.2 Supervised Machine Learning

Machine Learning involves the development of algorithms designed for computer learning, focusing on identifying regularities or patterns within data. In the realm of supervised machine learning, algorithms are designed to create functions that establish connections between input data and the desired output data. One specific application of supervised machine learning is classification. In classification tasks, algorithms such as decision tree classifiers or random forest classifiers are used to understand and learn the relationships between input and output data (Nasteski, 2017).

The algorithm identifies patterns within the data and generates models, which are then tested on new data to extract and predict information. This section delves into the principles underlying the decision tree classifier and the random forest classifier. Subsequently, it introduces the performance metric designed to evaluate these two algorithms in the classification process.

The decision tree classifier is an algorithm, comprising nodes, branches, and leaves. According to Sharma and Kumar (2016) in a decision tree each node represents a test on a selected feature, and the node's outputs are determined by branches defining specific conditions, that decide how the input of the node is tested. The output results in a leaf node representing a class label or in a further node where further tests are conducted on the remaining data. The primary objective of a decision tree is to predict the value of a target based on given inputs. To determine the first node and its corresponding split condition the decision tree classifier utilizes feature selection measurements, including entropy and information gain. Entropy, in this context, quantifies the impurity of a given dataset. The calculation involves the use of proportions (pi) representing the occurrence of a certain class within the dataset divided by the total count of classes within the dataset. The sum indicates the occurrence of the other classes within the dataset, for which the same formula is applied.

$$Entropy(D) = \sum -pi\ log2(pi) \qquad \textit{Eq. 3}$$

The information gain is the expected reduction in the entropy caused by the splitting of a node by a feature.

$$Gain(D, A) = Entropy(D) - \sum \frac{Dj}{D} \; Entropy(Dj) \qquad \textit{Eq. 4}$$

This formular represents D to be the entire dataset, A refers to the selected feature and Dj corresponds to the instances count in the features dataset. In the thesis, the dataset undergoes a process where it is divided into subsets based on specific ranges of values associated with the selected feature. Within each of these subsets, the count of instances, denoted as Dj in the formula, is determined. Dj signifies how many data points fall within the specified range of values for the chosen feature within a subset. Following this, the entropy for each subset is computed using the standard entropy formula, considering the proportions of different classes within that subset. The information gain for the selected feature is computed using the formula. This entire process is repeated for each feature, and the feature that results in the highest information gain is chosen to be the root node for the first split in the decision tree (Sharma & Kumar, 2016). According to Hegelich (2016), decision trees are susceptible to overfitting. An overfitted model tends to memorize the training data rather than learning the underlying patterns. As a result, the overfitted models may exhibit high performance on the training data set, but not on unseen data (Hegelich, 2016). The Random Forest algorithm is an improved version of the decision tree method, addressing the limitations of decision trees by taking advantage of an ensemble of trees. Unlike a decision tree, which builds up and predicts based on a single tree, the Random Forest Classifier builds up and predicts on multiple trees. The algorithm creates a group of decision trees, built up from a random selection of subsets of the training data. Here, an instance is selected multiple times, meaning that an instance appears more than once in the ensemble of decision trees, contributing to the diversity among the trees (Breimann, 2001).

Furthermore, the features describing the data are selected randomly for each split in one decision tree within the entire tree ensemble. The size of the random sample, as mentioned by Hegelich (2016), is set to the square root of the total number of features. At each split, the decision tree considers a subset of features rather than all of them. The rationale behind this is to reduce the correlation between the trees in the ensemble. By using a random subset of features, each tree in the Random Forest is exposed to different aspects of the data, leading to a more diverse ensemble. A high number of

features guarantees that all data points are evaluated by enough decision trees. After the algorithm has been trained on the training data and hence has built various decision trees, the entire model is constructed by combining the predictions of the individual trees. The prediction, that receives most of the votes, is chosen to be the final prediction of a given instance (Hegelich, 2016).

2.4.3 Unsupervised Machine Learning

Clustering refers to the problem of finding regions within a dataset where the objects exhibit higher similarity compared to clusters in different regions. Gan et al. (2020) differentiate between hierarchical, fuzzy, center-based, search-based, graph-based, grid-based, density-based, model-based, subspace, and miscellaneous clustering algorithms (Gan et al., 2020). The clustering process using the clustering algorithm density-based spatial clustering of applications with noise DBSCAN is employed to eliminate data points associated with a specific instance of interest.

According to Hahsler et. al (2019), the concept behind DBSCAN is the principle, that points are grouped into the same cluster based on their density-reachability to one another. The algorithm defines two parameters minimum Points *minPts* and epsilon *eps*. Minimum Points corresponds to the minimum number of data points within the neighborhood, whose size is defined by the radius *eps*. Leveraging these parameters, the algorithm identifies dense regions and categorizes data points within the region into core, border and noise points based on the following conditions (Hahsler et al., 2019).

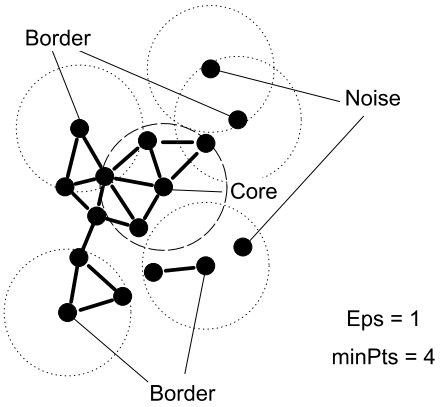


Figure 3: Clustering of the points using DBSCAN method: Core, Border, and Noise (Hahsler, Piekenbrock, & Doran, 2019).

A core point is identified when the number of datapoints within the defined distances meets or exceeds the minimum specified by *minPts*. A point situated within the distance, defined by *eps*, of a core point, yet lacking sufficient neighbors to be classified a core point itself, is designated a border point. Both core and border points contribute to the cluster, whereas points failing to meet the minimum criteria are labeled as noise or outliers. Applying the DBSCAN assumes estimating the parameters *minPts* and *eps* (Hahsler et al., 2019) Hahsler et al. (2019) state an approach for determining *minPts* by recommending a minimum value equal to the number of dimensions in the dataset plus one. In the context of the provided dataset, with xyz coordinates, indicating three dimensions of the dataset, the parameter *minPts* is set equal to four. To determine a value for *eps*, the manual modification of the parameters is required to eliminate datapoints associated with an instance of interest (Hahsler et al., 2019).

2.5 Geometric Model Reconstruction

Borrmann et al. (2015) describe two distinct techniques for the reconstruction of solid models: implicit representation and explicit representation. Implicit approaches involve Constructive Solid Representation (CSG), while explicit approaches use Boundary Representation (B-Rep). Each method has its own set of advantages and disadvantages. The implicit approach, utilizing CSG, allows for the traceability of modeling operations, facilitating the modification of geometric results by altering construction steps. CSG stores the parameters defining the shape of the solid and the operations to combine them together. On the other hand, geometric models constructed through explicit approaches (B-Rep) are challenging to modify due to an undocumented generation process. The B-Rep modeling method only stores the result of the operations that define the geometric shape. In summary, the implicit approach offers advantages in terms of simplicity, traceability, and modification. On the other hand, the explicit approach involves specifying geometric elements, providing a more detailed but less flexible representation. Eastman et al. (2011) state that these two methods, CSG and B-Rep, were in competition until they were integrated. Consequently, parametric modeling tools now accommodate both approaches. Through this integration, CSG is employed for geometry editing, while the B-Rep methodology is utilized for visualization, measurement, and clash detection purposes (Eastman, 2011). The subsequent section aims to explore techniques of solid modeling, examining both implicit and explicit approaches and parametric modeling.

2.5.1 Implicit Representation

Constructive Solid Geometry (CSG) relies on the representation of regular primitives and set operations such as union, intersection, and difference. However, its application is constrained by the availability of solid primitives (Shapiro, 2002). Voelcker and Requicha (1977) emphasize that primitive half-spaces form the lowest-level entities, combining to create instances of solid primitives. For instance, a system incorporating only cylindrical and planar half-spaces would return two solid primitives: a block defined by six planar half-spaces and a cylinder defined by a cylindrical and two planar half-spaces (Voelcker & Requicha, 1977). These primitives are combined using boolean operations to construct user-defined geometries. The implicit representation in CSG captures the entire modeling process within a construction tree, enabling the modification of construction steps. Voelcker and Requicha (1977) elaborate on the characteristics of the construction tree. Each node in the tree signifies a primitive or an operation, and the leaves denote instantiated primitive solids (Voelcker & Requicha, 1977). The accompanying Figure 4 illustrates the hierarchical structure in the geometric model representation using CSG.

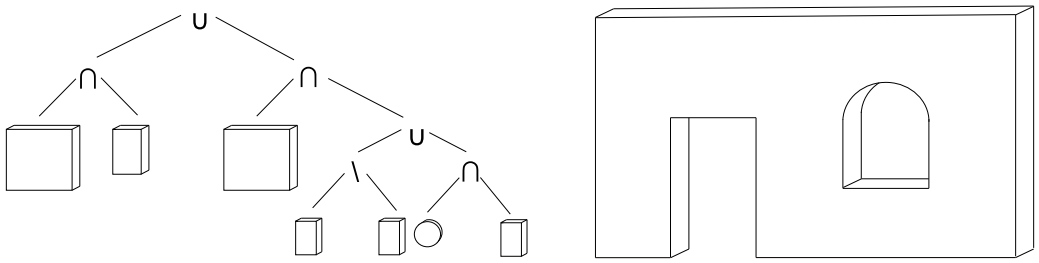


Figure 4: Constructive Solid Geometry. Construction tree and result (Borrmann et. al, 2015).

2.5.2 Explicit Representation

Boundary Representation (B-Rep) models represent solid objects by decomposing them into surfaces. These surfaces, in turn, are further divided by a set of faces. Each face is characterized by a collection of edges, which are represented as curves defined by pairs of vertices. This hierarchical structure provides a detailed way to represent complex solid geometries in B-Rep models. (Shah & Martti, 1995).

In the realm of geometry representation, Shah and Martti (1995) propose the following data structures: the polygon-based boundary data structure and the vertex-based boundary data structure. Oriented towards geometries characterized by planar surfaces with straight edges, the polygon-based boundary data structure represents each face as a polygon computed from its corresponding vertex coordinates (Shah & Martti, 1995). According to Borrmann et al. (2015) the vertex-based boundary data structure focuses on specifying the geometry through individual vertices in space. The shape of the solid is determined by the connection and position of these vertices. The data structure is utilized to represent simple solids without any voids. To achieve a geometric model with voids, the data structure must be expanded (Borrmann et al., 2015).

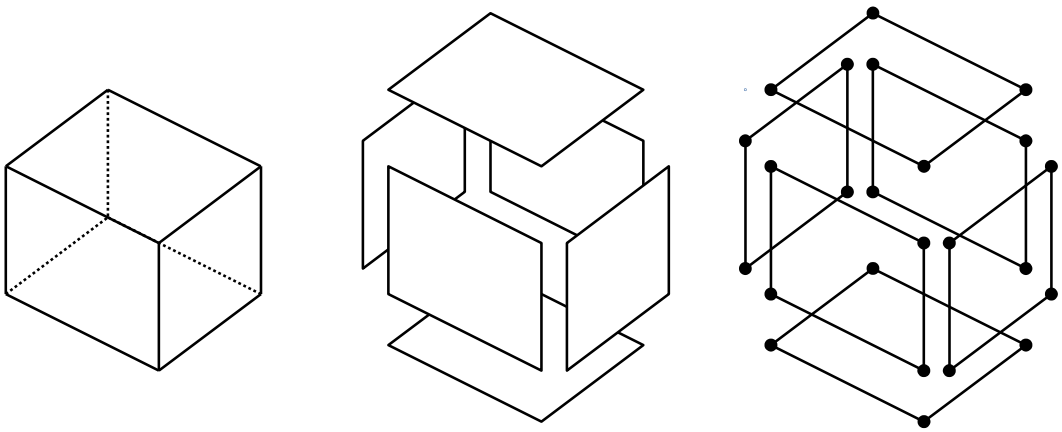


Figure 5: Boundary Representation by dividing a solid into its faces and lines defined by start and end vertex (Shah & Martti, 1995).

2.5.3 Parametric Modeling

Borrmann et al. (2015) define that the parametric modeling refers to the modeling of a geometric object with geometric and dimensional constraints. By changing parameters referring to measurements, the entire model is updated (Borrmann et. al., 2015). Eastman et al. (2011) distinguishes between three parametric relations, defining constraints between the parameters, that are described by geometric constraints like distances and angles, dimensional or descriptive relations like parallelism and verticality and equational relations between parameters (Eastman, 2011). Geometry-based parametric modeling involves creating geometries based on parametric rules. For instance, a sketch is generated with specific dimensional, equational, and geometric constraints. This sketch is then combined with a procedural geometric description, such as extrusion, to form a three-dimensional geometry. This method is considered implicit representation, because the entire construction history is stored, allowing for modifications at each construction step by adjusting parameters. In contrast, parametric BIM modeling puts constraints on the flexibility of parametric modeling due to predefined object types and constraints within the BIM application. The parametric approach in BIM is integrated on two levels: Once to establish the geometric construction objects like walls and stairs, and secondly, to define their positions within the overall building complex. As the object's position within the building complex must be defined, this approach typically includes predefined constraints between object types. These parameters encompass considerations like parallelism, orthogonality, alignment, distance, and identical measurements between objects (Borrmann et al., 2015).

The objective of the thesis is to generate a parametric staircase using PCD. The methodology involves using the parametric modeling capabilities of Autodesk Revit, a BIM application software. Revit facilitates parametric BIM modeling through the integration of built-in families that contain predefined parameters and constraints. The predefined staircase structure serves as a foundation for achieving parametric modeling. Revit facilitates parametric BIM modeling through the integration of built-in families that contain predefined parameters and constraints.

2.6 Research Gap

It is commonly agreed that a DT goes beyond a static model by incorporating real-time data for monitoring and operation. Scan-to-BIM involves using PCD to recreate a BIM that accurately represents real-world conditions in the built environment. This process is known for its labor-intensive and error-prone nature since the reconstruction process depends on the structure's size and complexity as well as the presence of building elements and construction details. To address the challenges within scan-to-BIM, AI is leveraged in automating the scan-to-BIM process. PCD is favored for its accuracy in capturing structural measurements, providing detailed as-built information. Geometric features describe the geometry of PCD by utilizing the eigenvalues of the covariance matrix. The information represents the staircase's geometry. The primary goal is to establish a classification model that identifies patterns within the data structure containing coordinate values and geometric feature values. This classification model groups points sharing similar values in coordinates and geometric features, classifying points within a point cloud corresponding to the staircase instance. Specifically for staircase structures, mainly defined by the number of steps and landings within its structure it becomes crucial to identify those and their dimensions. To achieve this several already established works are taken as a reference. Awwad et al., (2010) utilized the RANSAC algorithm in their study, applying it to PCD to identify planar surfaces within staircase structures. Their research aimed to enhance segmentation outcomes by refining the RANSAC algorithm (Awwad et al., 2010). Although the improved algorithm demonstrated better results, it was reduced to staircase structures without an emphasis on their geometric information. Tovari and Pfeifer (2005) proposed a segmentation method for ALS data based on region growing. Their approach involved estimating normal vectors at each point through k-nearest neighbors, and during the growing phase, neighboring points were incorporated into the segment based on criteria such as similarity in normal vectors, distance to the growing plane and distance to the current point (Tóvári & Pfeifer, 2005). Mehranfar et al., (2022) introduced a methodology aimed at extracting both geometric and semantic information form objects to improve the creation of a DT. The focus has been on enhancing algorithms for specific building objects (Mehranfar et al., 2022).

However, the previous mentioned works only consider improved algorithms that are applicable for certain building objects. In order to develop a workflow for the DT representation of a staircase structure, the ideas within the literature review are taken as a reference, representing central areas the thesis aims to resolve. These include:

-Interpretation of Classification Results: The section reveals an ongoing question concerning the accurate interpretation of results derived from supervised machine learning algorithms when applied to staircase classification.

-Selection of algorithms and parameters: The thesis aims to choose the most promising algorithm and its corresponding parameters, representing the need to determine the optimal approach for staircase modification.

-Geometric and Dimensional Constraints for Parametric Modeling: The literature highlights a particular research gap regarding the identification of appropriate geometric and dimensional constraints for remodeling the parameters of a staircase geometry within the context of scan-to-BIM processes.

3 Methodology

The current section outlines the method developed for the automatic creation of a digital staircase model from PCD. The process initiates with the utilization of CloudCompare and the programming language Python for the processing of point clouds. The initial objective involves the identification of staircases within the point cloud dataset. To achieve this, machine learning algorithms, particularly the Random Forest Classifier, are utilized for staircase identification.

Following the identification of staircases, adjustments are applied to streamline the representation of point clouds and prepare them for analysis. Planar surface detection within the point cloud is a major part to facilitate the subsequent analysis. The point cloud is divided into different parts based on the positions of planar surfaces.

The ongoing analysis of the point cloud's values involves the derivation of the coordinate's maximum and minimum information. Basic calculations are applied to extract width, length, and height values for each stair run. To achieve the number of steps, the current approach involves grouping points with similar z-coordinates. Calculated values for each stair run are presently being utilized to create the DT of the point cloud, specifically within the BIM authoring tool Autodesk Revit, which provides predefined parametric models for staircase components.

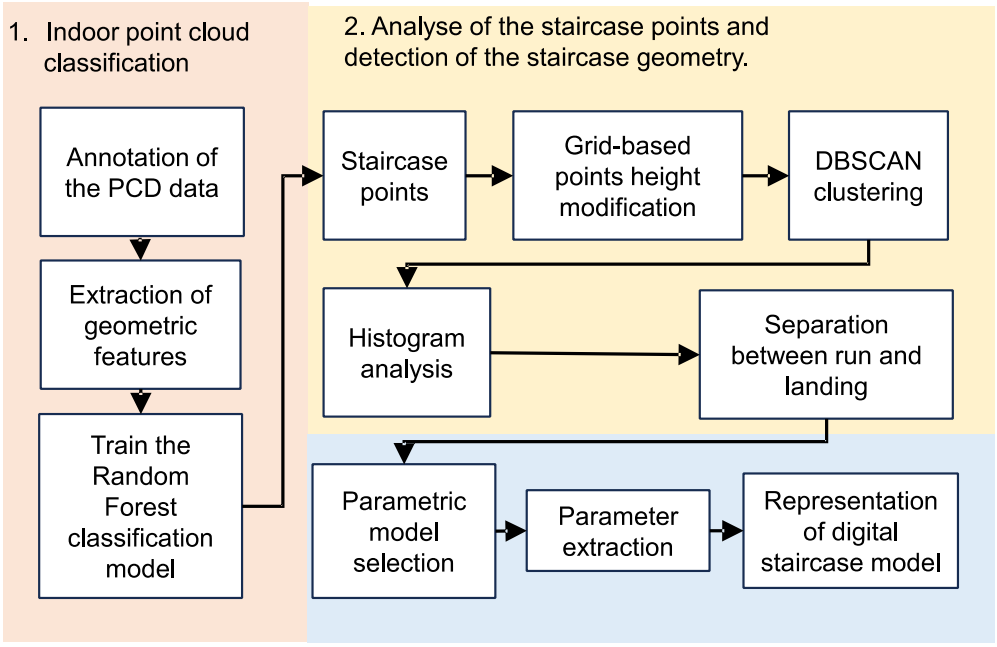


Figure 6: The proposed workflow for the creation of a digital staircase model using PCD.

3.1 Proposed Workflow

For evaluation the data-driven bottom-up approach is centered on staircase instances captured by a TLS at the Technical University of Munich. It presents three different point clouds with staircases, each characterized by a similar geometry involving landings and runs perpendicular to each other. Consequently, the introduced bottom-up approach is adaptable and is applied to staircases from various environments, provided they adhere to the conditions of perpendicularity and planar surface interruption.

3.2 Semantic Enrichment

The initial phase revolves around identifying staircases within indoor point clouds. To achieve this, machine learning algorithms are utilized to establish a tool capable of assigning labels to individual points within the dataset. This entails the utilization of a Random Forest Classification Model. In essence, the model undergoes training on existing datasets that have already been labeled. After training the model, it is then applied to unclassified datasets, assigning labels to points associated with staircase geometries. For training purposes, a diverse set of point clouds from various sources is used. These datasets are labeled using domain knowledge within the open-source software CloudCompare. The labeling process encompasses distinct categories, namely Wall, Ceiling, Floor, Furniture, Stair, and Column.

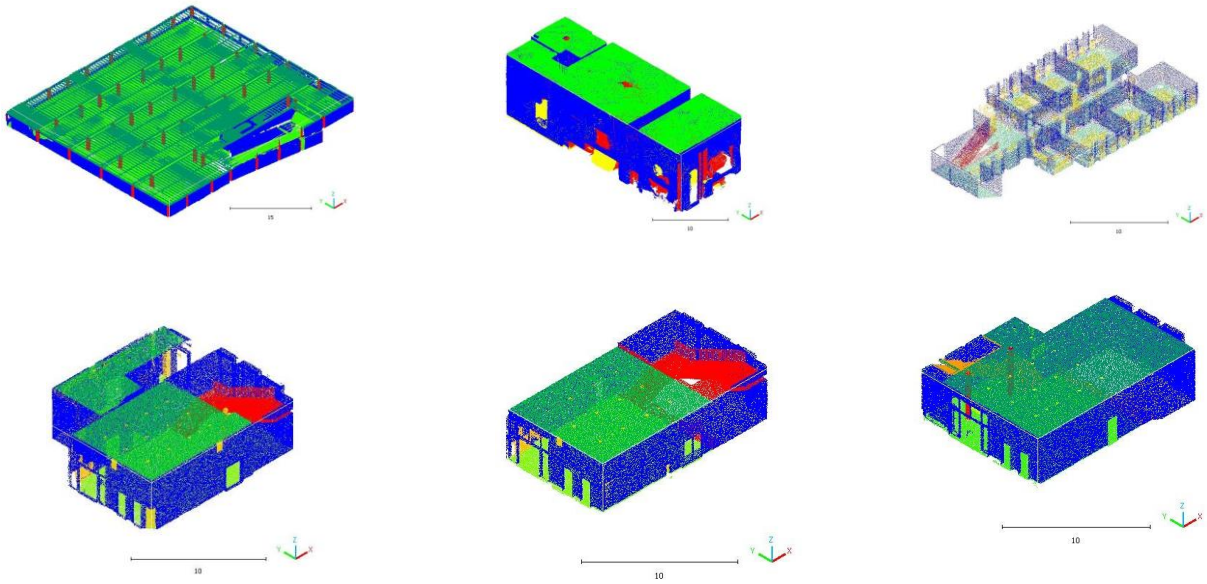


Figure 7: Training Datasets

3.2.1 Feature Extraction

Each point cloud undergoes an analysis to compute geometric features that characterize the overall geometric shape within the dataset. These geometric features include verticality, planarity, surface variation, omnivariance, anisotropy, as well as the calculation of normals with reference to the point cloud's barycenter and their relative height. The neighboring distance represented by the radius contains the necessary points that are included for feature calculation. The larger the radius, the more points are included within the search area. The size is necessary to consider relevant geometries within the same neighborhood. For instance, larger search areas capture more complex geometric structures like furniture. Geometric features describe the point clouds geometric characteristics. Most of the geometric features are calculated by using the eigenvalues of eigenvectors derived from the covariance matrix of points located within the predefined neighborhood of the point within the PCD. The covariance matrix includes the calculation of the data’s variance, which indicates how much the data points are concentrated around the mean of each axis. Based on the covariance matrix of the PCD, the eigenvectors and their corresponding eigenvalues are calculated. Depending on the size of the covariance, which is in the case of point clouds a 3x3 matrix, 3 eigenvalues are calculated. The eigenvectors represent the direction of the computed coordinate system at the given neighborhood and the eigenvalues indicate the magnitude of the variance in the direction of the eigenvectors (Lu & Yang, 2019). The eigenvalues are necessary to compute most of the geometric features like planarity, omnivariance, surface variation of each point in a point cloud. According to He, et al. (2022), omnivariance relates to the surface’s degree of fluctuation. A higher omnivariance value represents a more complex surface within the neighborhood. This measure is useful for distinguishing different types of surfaces. Points with higher omnivariance may correspond to areas with more irregular surfaces, while lower omnivariance may indicate regular surfaces like planar surfaces. The formula for surface variance represented in Table 1. Surface Variance is established by the ratio between the lowest eigenvalue λ3 and the sum of all eigenvalues. A high value indicates more variation within the neighborhood. A low value refers to a planar surface within the neighborhood. Surface variation focuses on local variability within a neighborhood, while omnivariance focuses on the overall variation across the entire surface (He, et al., 2022).

Eigenentropy is based on the concept of entropy, which measures the disorder in a dataset. In this context, eigenentropy is calculated using eigenvalues. The calculation results in a measure of diversity of the eigenvalues. The more the eigenvalues differ, the more diverse they are, resulting in a higher value for eigenentropy. The more equal the eigenvalues are, the lower the eigenentropy. Anisotropy calculates how features or measurements change to different directions in space. A high anisotropy value indicates a greater elongation along the axis determined by the eigenvector associated with the largest eigenvalue λ1 (Hackel et al., 2016). Planarity describes how flat a surface is by defining a plane within the neighborhood area. A high value indicates a flat surface. Depending on the computation, the verticality feature is calculated using different formulas. The feature is useful for distinguishing between horizontal and vertical planar surfaces. A higher value indicates that the point is associated with a vertical surface. To calculate points that belong to certain planar surfaces of the coordinate system, which are rather parallel to the z-y-plane or to the z-x plane, the y and x-coordinate of the normal vector is extracted. In conclusion, by extracting the geometric features of Nz, which is set in reference to the geometric feature of verticality, Ny and Nx, planar surfaces that are parallel to the planar surfaces of the coordinate system are recognized. The verticality formula described in the previous section refers to the normal vector of the point. The absolute value of the z component of the normal vector is subtracted from one (He, et al., 2022).

Omnivariance	$(\lambda 1 \cdot \lambda 2 \cdot \lambda 3)^{\frac{1}{3}}$
Surface Variation	$\frac{\lambda 3}{(\lambda 1 + \lambda 2 + \lambda 3)}$
Eigenentropy	$-\sum_{i=1}^{3} \lambda i \cdot ln(\lambda i)$
Anisotropy	$\frac{(\lambda 1 - \lambda 3)}{\lambda 1}$
Planarity	$\frac{(\lambda 2 - \lambda 3)}{\lambda 1}$
Verticality	$1 - \lvert Nz \rvert$

Table 1: Geometric Features formulas with the eigenvalues λ1, λ2, λ3 (Hackel et al., 2016).

The verticality feature considers the absolute value of the normal vectors, thus equalizing negative and positive values. To separate negative and positive normals to relate them to certain classes, the computation of the normals is processed differently. The average location of the points in the point cloud is described by the barycenter. The normal vectors orientation of the points of the point cloud can be oriented either positively or negatively with reference to the barycenter. A negative barycenter indicates that the normal vector points towards the direction of the center. A positive barycenter indicates that the normal vectors point away from the barycenter. In the case of the commonly used positive barycenter, ceiling points are represented positive, because its computed positive z component corresponds to the direction of the positive vector from the cloud barycenter. Points belonging to the floor are described by a negative normal vector, because its normal vector directs towards the barycenter and thus its negative z component does not correspond to the direction of the positive vector from the cloud barycenter.

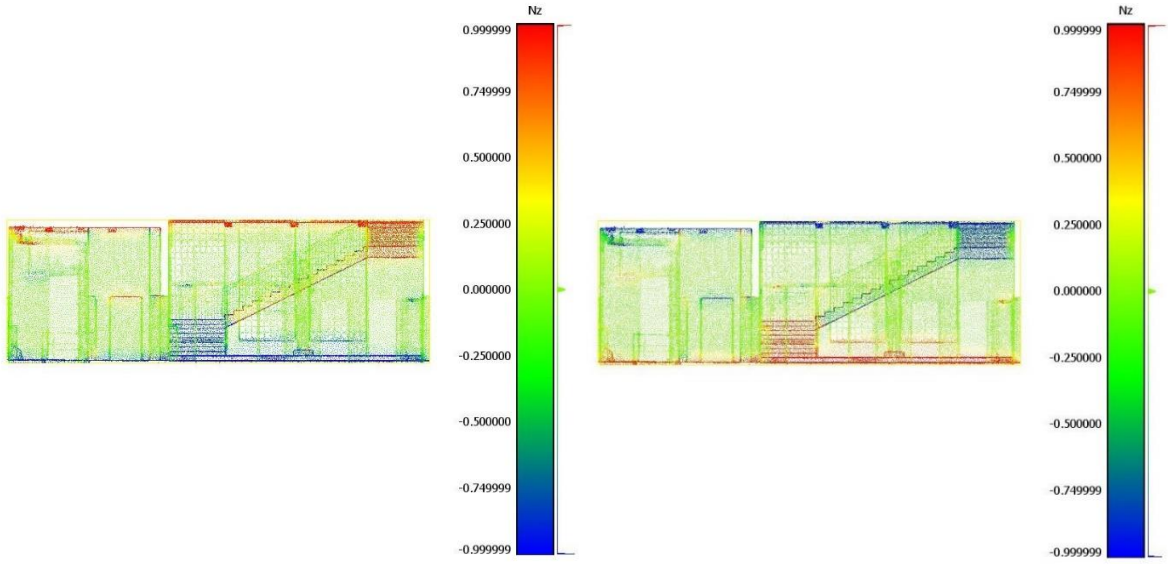


Figure 8: Normal Vectors according to the barycenter.

Left: Normal Vector Orientation according to positive barycenter

Right: Normal Vector Orientation according to negative barycenter

Using normal vectors with reference to an orientation creates specific information about vertical and horizontal surfaces. In the previous example, surfaces with normal vectors that are aligned to the z-axis have been separated. The same method is used to distinguish between surfaces with normal vectors that are aligned to the y-axis or x-axis. The normal vector is an additional feature in the classification model, which increases the prediction accuracy of the classes floor and ceiling. To enhance the discrimination between floors and ceiling, an additional feature is introduced. For each point, within a neighborhood defined by a specified radius, the maximum and minimum z-coordinates are computed. The algorithm distinguishes flat surfaces associated with ceiling or floors by evaluating whether the difference between a point's z-coordinate and the minimum or maximum z-coordinate within the neighborhood is smaller than a given tolerance. A true condition yields a value. In the case of a ceiling, the algorithm detects flat surfaces by comparing a point's z-coordinate with the maximum z-coordinate within its neighborhood. This condition is fulfilled, because the maximum z-coordinate of points within the neighborhood is similar to that of the given point. Conversely, for floors, the algorithm distinguishes flat surfaces by evaluating the difference between a point's z-coordinate and the minimum z-coordinate within its neighborhood. Here, the condition is fulfilled, because the z-coordinate of points within the neighborhood is similar to that of the given point. In summary, the geometric features derived from the PCD encloses its geometry. The computation of these features considers elements such as normal vectors, coordinates, eigenvalues, and eigenvectors. The selection of geometric features is central in distinguishing between elements.

Following the calculation process, the PCD, initially represented by seven columns of information, is transformed into a dataset enriched with 14 information columns. The identical procedure for feature calculation is implemented on the unlabeled dataset, resulting in the generation of 13 information columns. The missing column refers to the point cloud's label, which will be determined by the trained random forest classifier.

3.2.2 Training the Random Forest Model

To identify points associated with the label stair within the three introduced point clouds, the classification model is trained by four to five datasets out of the datasets represented in Figure 7. Every training dataset is composed of 14 information columns. In contrast, the target point cloud, for which the label is to be determined, comprises 13 information columns. In total, four to five different point cloud datasets are used for building up the random forest classification model. By configuring the test size to be 0.2, the complete dataset undergoes a division into 80% training dataset and 20% testing dataset. The random forest classification model is based on 80% of the entire dataset, and its performance in assigning the correct label to each instance is evaluated using the remaining 20%. Once the machine has constructed the classification model, the evaluation of the model includes publishing performance metrics including accuracy, precision, recall, and F1-score.

Precision and recall are measures to evaluate the model's performance. A dataset with instances that rather belong to the class or not is represented. The algorithm calculates and predicts results, represented by the following Table 2.

	Predicted		
Actual		Class	Not Class
	Class	TP	FN
	Not Class	FP	TN

Table 2: Definition of TP, FP, FN and TN.

True Positive TP signifies the count of instances correctly predicted as a particular class, accurately belonging to that class. False Positive FP represents the count of instances predicted as the class, but do not belong to it. False Negatives FN denote instances predicted not to belong to the class but, in fact, do belong. True Negatives TN reflect instances predicted not to belong to the class, and indeed, do not. Precision serves as a metric quantifying the accuracy of positive predictions generated by the model. It is computed as the ratio of True Positive predictions to the sum of True Positives and False Positives. A high precision value indicates that the model has a high performance at predicting positive instances correctly (Jesse & Goadrich, 2006).

$$Precision = TP \,/\, (TP + FP) \qquad \textit{Eq. 5}$$

Recall, on the contrary, is calculated as the ratio of true positive predictions to the sum of true positives and false negatives. A high recall value indicates that the model is effective at reducing mispredictions and capturing most per positive instances.

$$Recall = TP \,/\, (TP + FN) \qquad \textit{Eq. 6}$$

The metric Intersection over Union (IoU) indicates how the model is performing in segmenting each class. It is commonly associated with object detection and segmentation tasks where bounding boxes or segmented regions are involved. In the context of the code, IoU is calculated based on the confusion matrix, by using the ratio of true positives and the sum of true positives, false positives, and false negatives for the class. In the context of the code, the metric IoU is used to assess the model's performance in predicting instances of different classes, considering the information available in the confusion matrix. It does not traditionally refer to object detection, but rather extends the concept to evaluate the spatial agreement between predicted and true instances based on class labels (Jesse & Goadrich, 2006).

$$IoU = TP/(TP + FP + FN) \qquad \textit{Eq. 7}$$

After training the Random Forest Classifier on a labeled dataset, the model is applied to predict the class labels of points within the unlabeled point cloud. The process entails calling the Random Forest Classifier's predict function, which then assigns a predicted class label to each instance in the point cloud based on the model's understanding of the input features. Upon applying the trained Random Forest Classifier to the PCD, a new file is generated, augmenting the dataset with an additional information column. This newly appended column represents the segmentation of the PCD.

3.3 3D Model Reconstruction

The subsequent process requires adjusting the configuration of the staircase to simplify its shape and prepare it for the forthcoming process. The detailed modification entails the elimination of points linked to the railing, shifting the staircase structure upwards, and the partitioning of the staircase into distinct sections based on the placement of landings within the staircase.

3.3.1 Bottom-up Parameter Extraction

An interest lies in isolating specific points associated with staircase geometries. Subsequently, a two-dimensional grid with a parametrically adjustable cell size is overlaid onto the XY plane, which has been previously projected. This grid partitions the dataset into cells, each containing a different number of point instances with XYZ data. The grid's objective is to ascertain the maximum z coordinate within each cell and assign this value to all points within the grid cell. This procedural adjustment is aimed at simplifying the point cloud's structure.

Additionally, the DBSCAN algorithm is applied on the point cloud. In this application, the epsilon parameter is manually set to 0.134, and the minimum samples are set to four. This choice aligns with the recommendation by Hahsler et al. (2019). Upon closer examination of the clustering results, specific clusters representing points corresponding to the railing are eliminated to achieve the intended outcome for further processing.

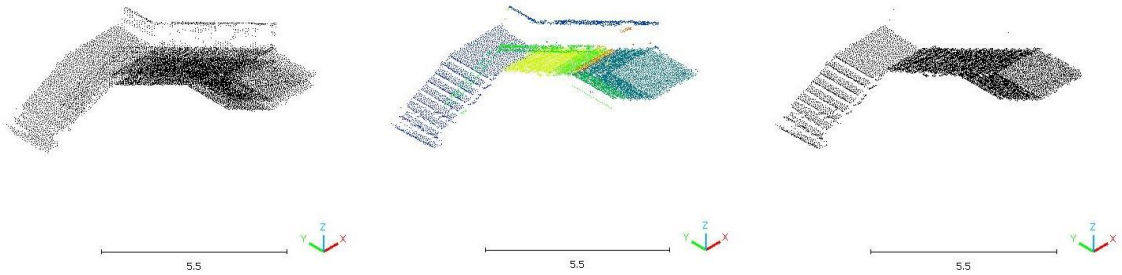


Figure 9: Left: Staircase Instance. Middle: DBSCAN Clustering result
Right: Result emerged from the combined application of the grid and the DBSCAN algorithm.

The subsequent step involves inspecting the z coordinates within the modified point cloud. Points associated with horizontal surfaces exhibit nearly identical z coordinates. Yet, it's crucial to note that these horizontal surfaces may also encompass steps within the PCD. To pinpoint the z coordinate values corresponding to landings within the PCD, the approach involves identifying the most frequently occurring z coordinates. This is accomplished by utilizing a histogram analysis and extracting the peak values discerned within the specified histogram. Points related to the initial landing all have a consistent z coordinate of around -14.62, while points associated with the second landing share a z coordinate value of -12.15. Following this, the PCD is divided into three segments. The first segment, denoted as part a, encompasses all points with a z coordinate equal to or less than -14.62. Part b consists of points characterized by z coordinates falling between -14.62 and -12.15. Lastly, part c comprises points with z coordinates greater than -12.15.

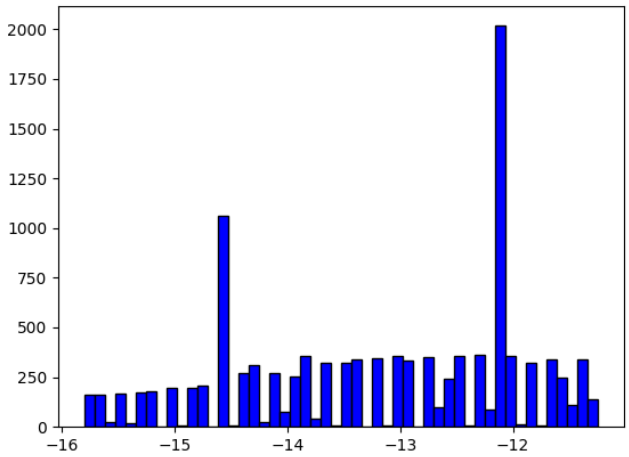


Figure 10: Histogram of the most frequent z-coordinates.

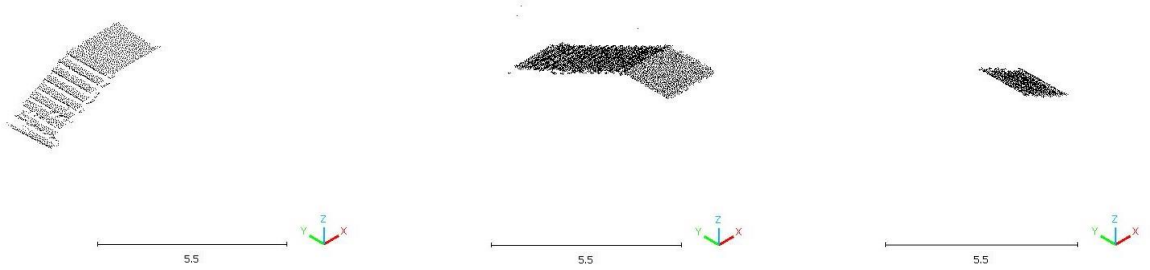


Figure 11: From left to right: the components of the staircase instance are organized as follows: Part a, Part b, and Part c.

The previous procedure ensures the extraction of parameters defining the geometric structure of the staircase. This section is dedicated to retrieving specific values such as height, width, length, and the count of steps for individual segments of the staircase. To obtain the parameters for height, width, and length, the process involves calculating the differences between the maximum and minimum coordinates within the according part of the staircase. Simultaneously, determining the number of steps entails grouping points with similar z coordinates and then quantifying the groups, as illustrated in Figure 12. The grouping of z coordinates is based on a threshold, that defines how much the z coordinates may differ in between to each other. In the according example, this threshold is set equal to 0.1625.

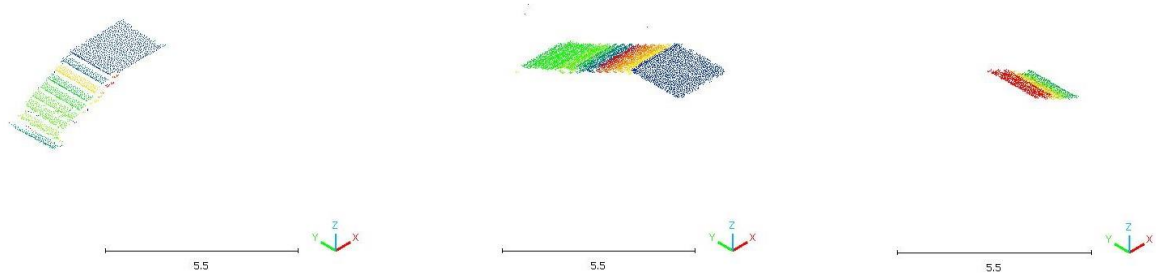


Figure 12: Result of the quantification of planar surfaces by grouping points with similar z-coordinates.

The determination of width and length parameters involves coordinate calculations for each part, given that the alignment requires the use of different y or x coordinates. In contrast, while the height parameter remains consistent for every part of the staircase due to its definition by the z-coordinate, a discrepancy arises as the points are shifted

upwards through grid application. This shift distorts the height value, no longer accurately representing the true height of the staircase. Applying a grid to the point cloud data results in retaining only the top surfaces of the staircase. The calculation of height involves identifying the minimum z-coordinate, which is found at the lowest surface of the first step. However, the actual minimum z-coordinate of the staircase begins with the riser height. To determine the correct height value, the calculation requires subtracting the maximum z-coordinate from the minimum z-coordinate while adding the average tread height for each part of the staircase.

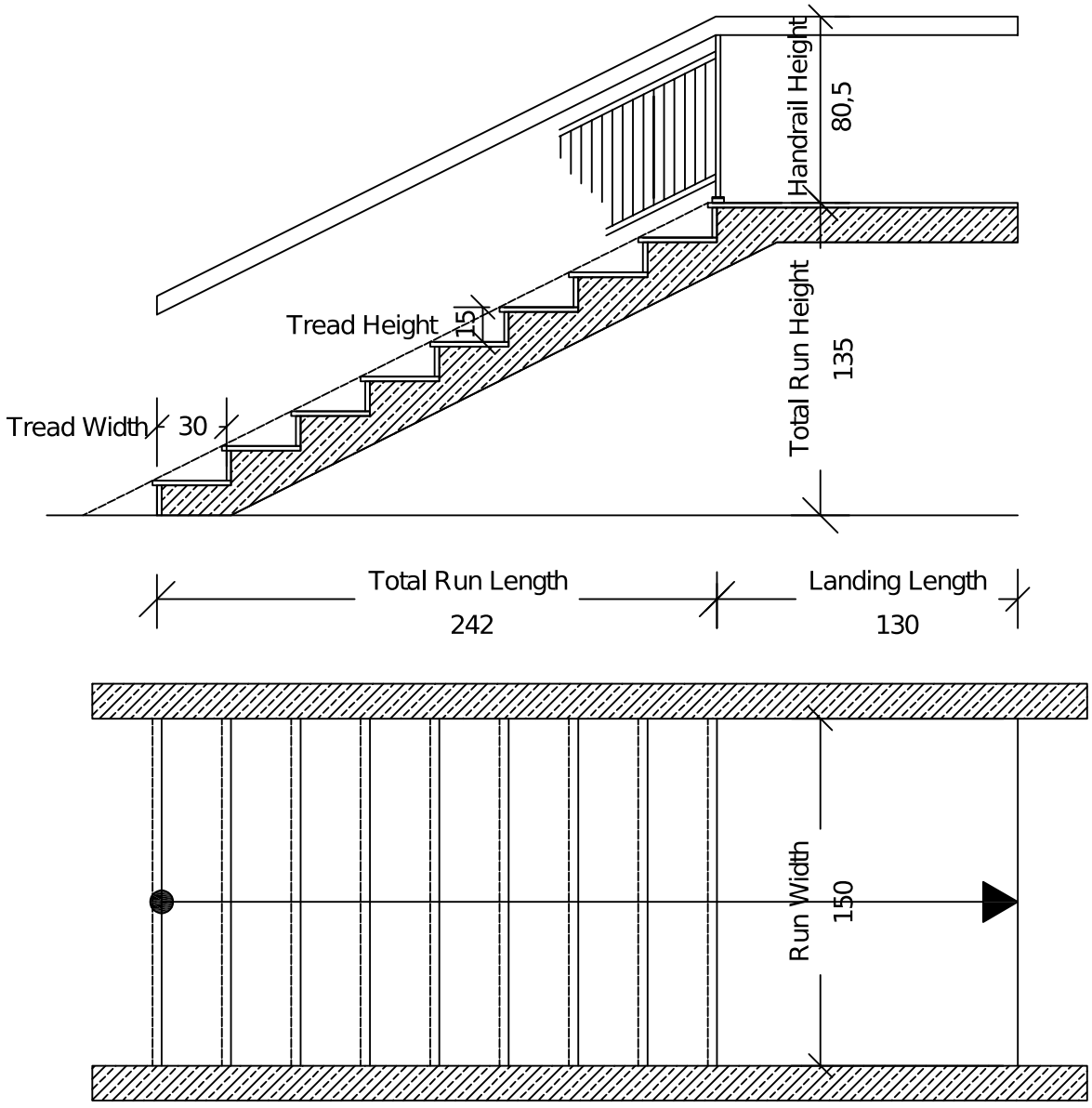


Figure 13: Staircase Parameters Tread Width, Tread Height, Handrail Heigth, Run Height, Run Length, Run Width and Landing Length.

	Run Width	Run Length	Run Height	Tread Heigth	Tread Depth
Part a	Max(y)-min(y)	Max(x)-min(x)-Width	Max(z)-min(z)+ tread height	Height/ Number of Steps	(Length -Width)/ Number of Steps
Part b	Max(x)-min(x)	Max(y)-min(y)-Width	Max(z)-min(z)+ tread height	Height/ Number of Steps	(Length -Width)/ Number of Steps
Part c	Max(y)-min(y)	Max(x)-min(x)	Max(z)-min(z)+ tread height	Height/ Number of Steps	Length/ Number of Steps

Table 3: Basic formulas for each part within the staircase.

3.3.2 Model Reconstruction

The values acquired previously are incorporated into the parametric model. While certain software offerings come with pre-existing parametric models, it is feasible to define one using the Python programming language. The next stage involves creating the staircase. First within the parametric models embedded within the staircase library and then within the pre-established parametric model in the BIM authoring software Autodesk Revit. The parametric modeling of staircase structures using the Python programming language encompasses 16 different designs, each varying in the number of landing treads and the directional alignment of the staircase. The structure is constructed by stacking cuboids, with the number of cuboids, defined as the "number of boxes" parameter, influencing the length of the runs - either extending or reducing it. The dimensions of the cuboids determine the tread depth, tread height, and total width of the staircase. Additionally, a parameter is introduced to specify the landing length of the last box in the structure, serving as the width value for the corresponding run.

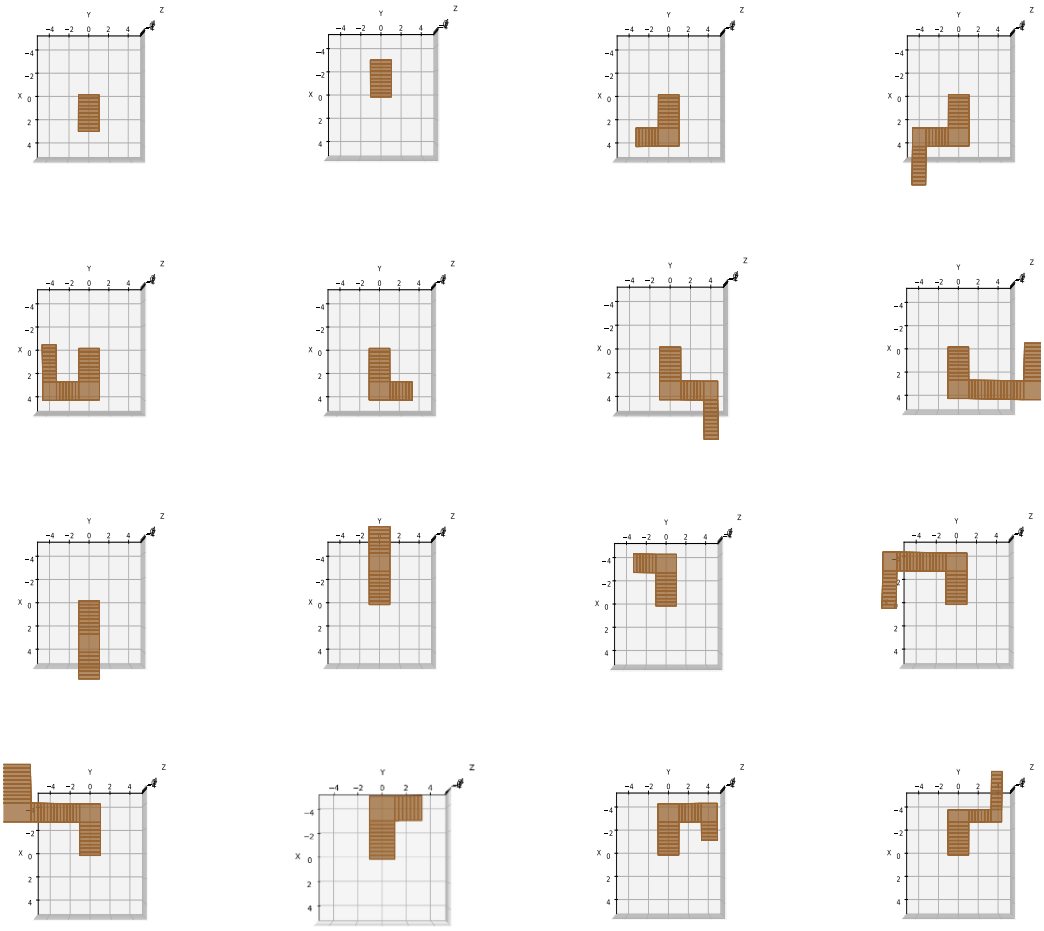


Figure 14: Different parametric staircase design.

In cases where two landings are present in the geometry, the landing length is incorporated twice in the parameter definition. Similarly, with three runs, three different values indicate the number of steps in the parametric model. This partitions the staircase into three sections. The values established earlier are incorporated into the chosen staircase. This is facilitated by the parametric model, which features two landings and aligns the runs in a manner consistent with the orientation observed in the point cloud data. The parametric model is generated by defining a series of cuboids. The starting center coordinates establish the position of the initial cuboid within the coordinate system, serving as the foundation for constructing the staircase's geometry. Modifying the coordinates of the initial center subsequently relocates the entire staircase, given that its shape is dependent on the first cuboid. The first staircase is composed of cuboids with uniform dimensions depth, width, and height, stacked atop one another. To determine the alignment of the cuboids in the coordinate system, the center point of each box must be adjusted as new cuboids are created. This adjustment involves updating the coordinates based on the previous cuboid; specifically, the x-coordinate is updated by adding the depth of the cuboid’s dimension to the initial center’s x- coordinate, while the y-coordinate remains constant. The z-coordinate is determined by adding the z-coordinate of the initial location to the height of the cuboids' dimensions. The termination of the staircase is marked by a landing, which is also constructed as a cuboid but with distinct dimensions. While the width and height remain consistent, a new parameter, landing_length, is introduced to specify the depth of the landing. Consequently, the landing's dimensions are determined by landing_length, width, and height. To establish the location center of the landing, a two-step process is employed. Firstly, the location center point of the last box within the run is extracted. Subsequently, a new location center is defined based on this extracted point. This definition ensures that the landing maintains the same height and width as the last cuboid's center point, with only the x-coordinate being adjusted based on the newly introduced parameter, landing_length. Based on the center location of the landing box, the initial cuboid of the second staircase run is determined. This location center is established by taking the original coordinates of the landing's center. The dimensions of the cuboids in the second staircase do not equal the cuboid’s dimensions in the first staircase.

This difference arises because the orientation aligns with the y-axis, and the width of the boxes corresponds to the landing_length of the landing. Consequently, the dimensions of the cuboids in the second staircase are defined by landing_length, depth, and height. The definition of the staircase run is like that established in the first staircase run. The loop continues iterating until it has generated all the cuboids, and after each iteration, the location center of the cuboid is redefined.

Once again, the staircase concludes with a landing, marking the second landing in the entire structure. The location center of this landing is derived from the location center of the last cuboid within the second staircase run, and its dimensions closely align with those of the cuboids in the second stair run. The dimensions of the second landing are specified by landing_length, landing_length2, and height. Notably, landing_length2 is a newly introduced parameter, determining the y-coordinate of the landing box and serving as the only dimension that deviates from the dimensions of the cuboids in the previously defined stair run. The third run of stairs is constructed based on the location center of the second landing. Thus, the dimensions of the cuboids in this run are determined by depth, landing_length2, and height. The resulting staircase is illustrated in Figure 15, which is set in comparison with the initial point cloud data.

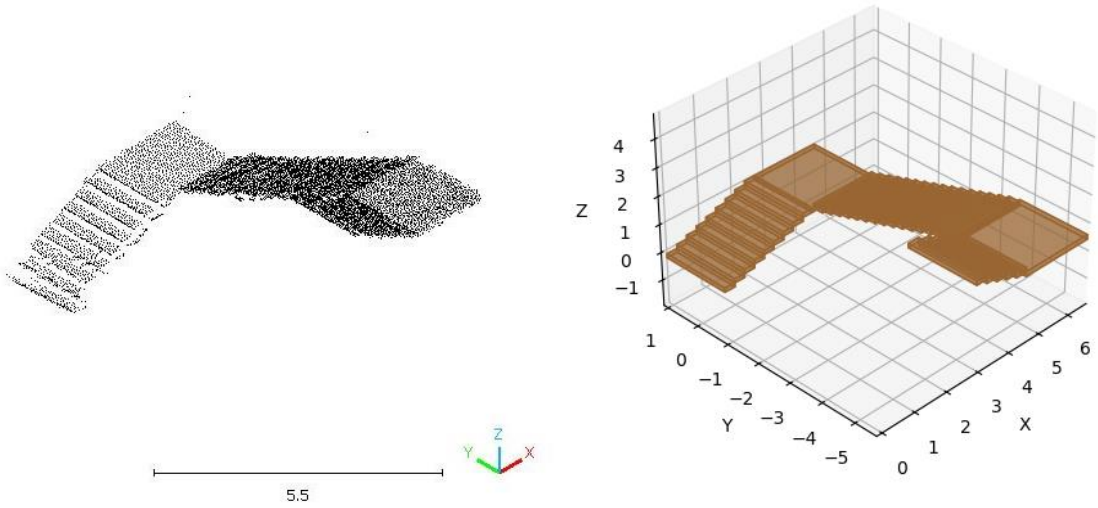


Figure 15: Comparison between point cloud data and parametric model.

num_boxes	9	height	0.14
num_boxes2	16	depth	0.30
num_boxes3	5	dimensions	(depth, landing_length, height)
landing_length	1.91	dimensions2	(landing_length1, depth, height)
landing_length1	1.86	dimensions3	(depth, landing_length2, height)
landing_length2	1.90	Initial_center	(0,0,0)

Table 4: Parameters defining the staircase.

The parametric definition of the staircase structure relies on specifying cuboids and their relative positions to each other. This approach enables the parametric determination of various staircase dimensions, including the number of steps, tread width, tread height, run width, run length, and overall run height.

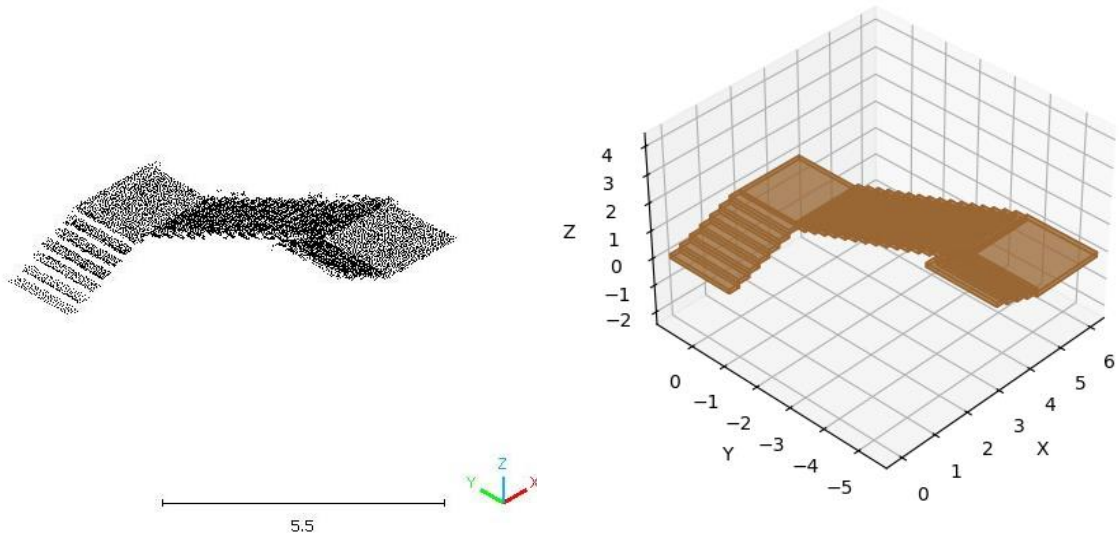


Figure 16: Left: TUM-F1, Right: Parametric Model.

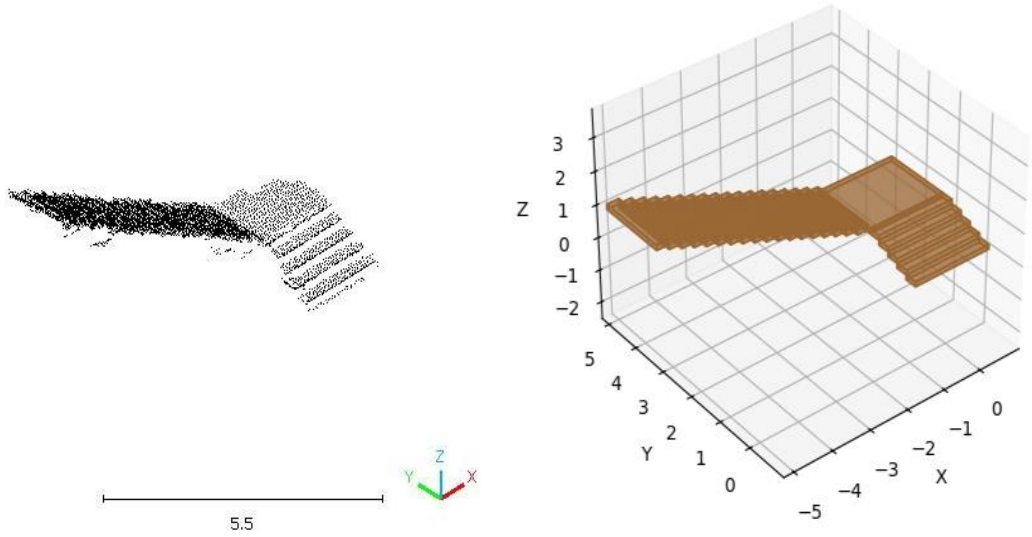


Figure 17: Left: TUM-F2, Right: Parametric Model.

Considering the established constraints, the staircase is modeled parametrically using Autodesk Revit. The subsequent content outlines the parametric modeling process applied to the extracted staircase. In Revit, there is a differentiation between two types of families: system families, which encompass construction elements like stairs, and loadable families, which typically include purchased items such as furniture. To derive a parametric model from PCD, the system family stair is utilized. In the context of a staircase structure, constraints are embedded within the system families of run, landing, and railing. Each of these system families contain constraints and parameters. The positioning of the staircase within the larger building complex is determined by establishing the height, which, in turn, depends on the vertical distance between two levels. This distance parameter is defined by a geometric constraint, as illustrated in Figure 18.

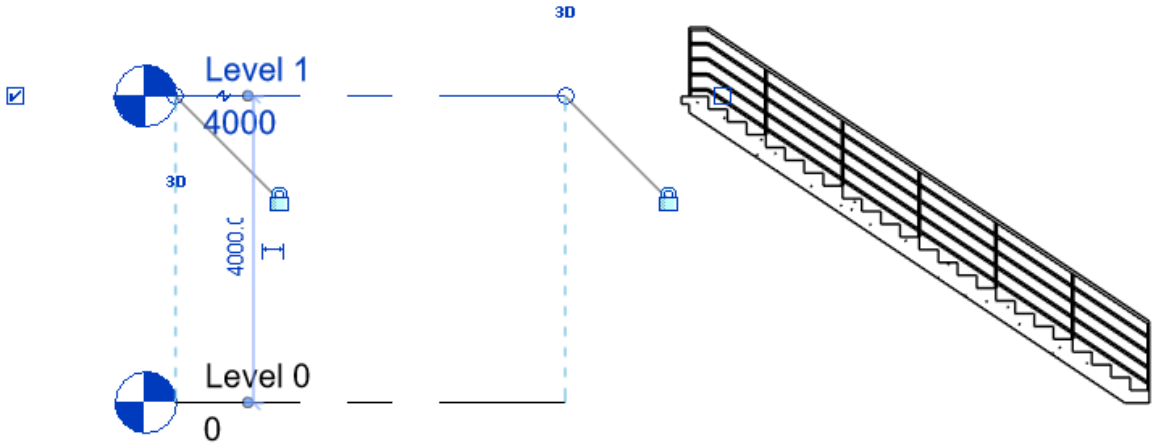


Figure 18: Geometric constraint representing the distance between two levels.

The constraint is embedded within the instance parameters of the staircase object, preventing its direct modification. This constraint depends on the predefined levels through a geometric constraint, making the modification of its value difficult. To specify the width of the staircase, it is necessary to access the parameters of the embedded system family "run" since the width is geometrically constrained between parallel boundary lines of the run. The length of the run is fixed and cannot be altered, as it depends on the distance between two levels and the number of risers. The number of risers is a significant parameter, as it determines the tread height through an equational constraint. The equational constraint arises from dividing the parameter obtained from the distance between two levels by the number of risers, resulting in the tread height. This establishes a relationship between the parameters, forming an equational constraint. The tread length, in turn, dictates the length of the runs. By adjusting the tread length, the total length of the staircase run is rather increased or decreased. Additional specifications are defined within the type properties of the staircase, as depicted in Figure 19. These properties set maximum and minimum values for riser height and tread depth. These values do not directly impact the constraints established in the instance properties. Despite the presence of an equational constraint suggested by Neufert's guidelines, linking maximum and minimum values for riser height and tread depth, this formula is primarily utilized for slope calculation. Neufert's guidelines propose a formula for calculating tread height and tread width (Neufert & Neufert, 2012):

$$2 * tread\ height\ +\ tread\ depth\ =\ 59 - 65\ cm \qquad Eq.\ 8$$

While this formula is embedded within the type properties, modifying its parameters only changes the numeric result without affecting the geometry of the staircase. However, once the minimum tread depth is established in the type properties, it serves as a threshold for the "Actual Tread Depth" parameter within the instance properties.

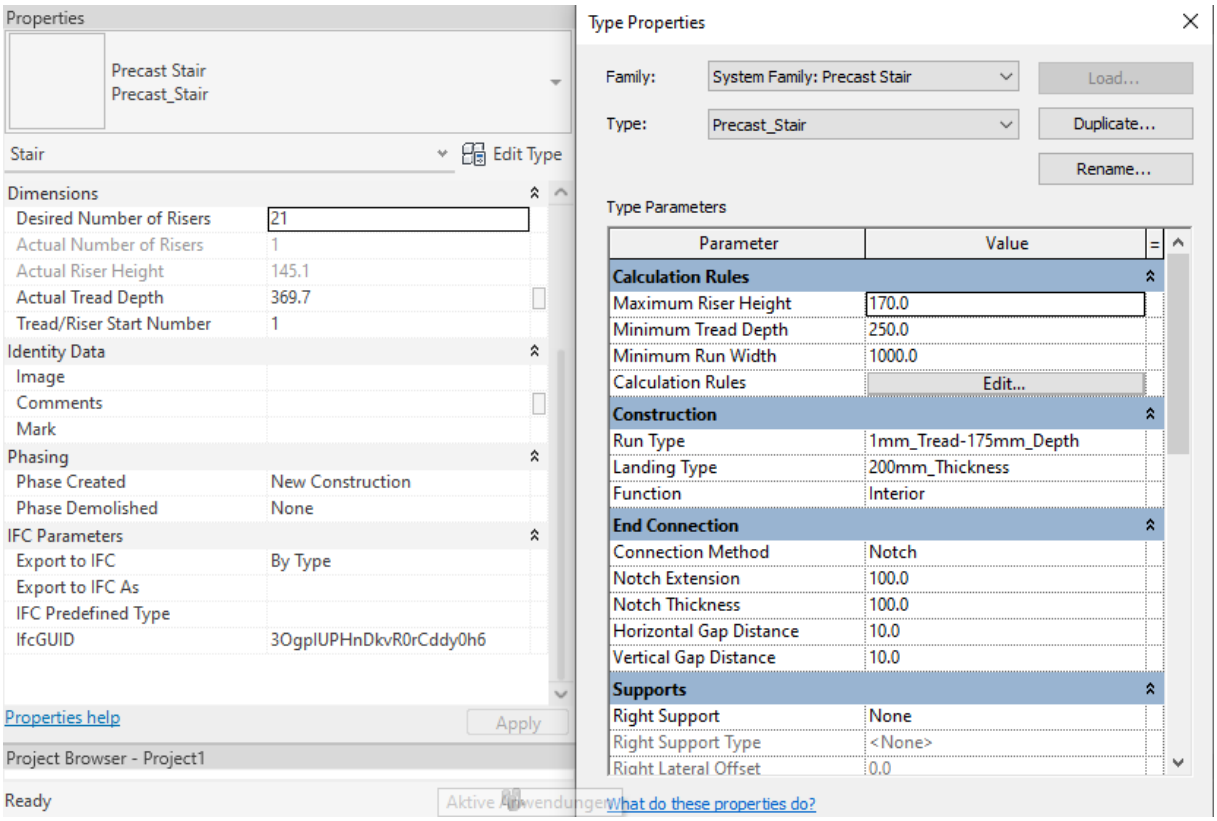


Figure 19: Instance properties of the built-in family stair on the left and type properties on the right.

In summary, the system family, comprising runs and landings, incorporates geometric and equational constraints within both instance and type properties. These constraints generate outcomes for various parameters that are not directly altered due to their interdependence with other parametric values (Duell et al., 2013).

4 Implementation and Results

To generate a parametric DT from the staircases represented within three datasets, the steps outlined in the methodology described in section 3 are applied.

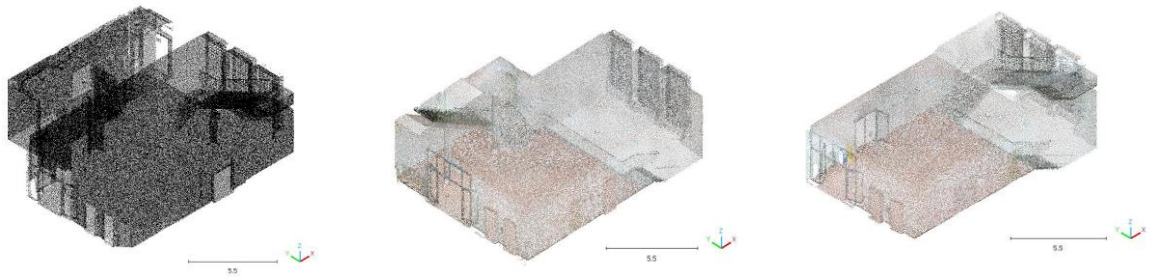


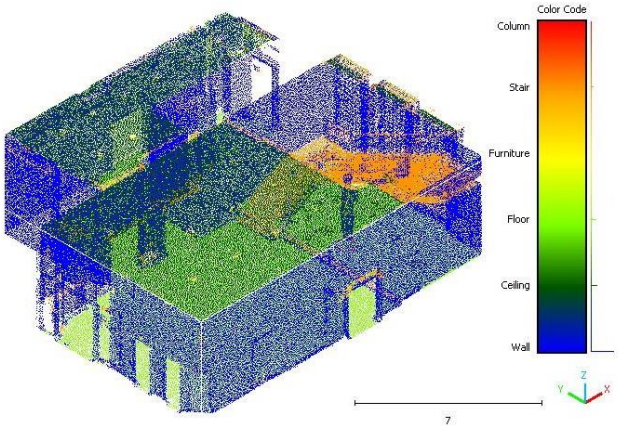
Figure 20: Point Cloud Data to be labeled by the classification model.

Acquired at the Technical University of Munich. From left to right: TUM-F0, TUM-F1 and TUM-F2.

Once the PCD is classified and the staircase has been defined, their coordinates undergo analysis to simplify the shape and isolate specific clustering groups. Following this simplification, parameters are extracted from the staircase, serving as the foundation for integration into the parametric model. The parameter extraction process may require code modification, as the staircases consist of a different number of runs and landings with a directional alignment that differs within the datasets. The section will present the results obtained through this methodology applied to various datasets. Despite variations in datasets, the staircases share similar characteristics, such as the perpendicularity between runs.

4.1 Experimental Result on the semantic enrichment

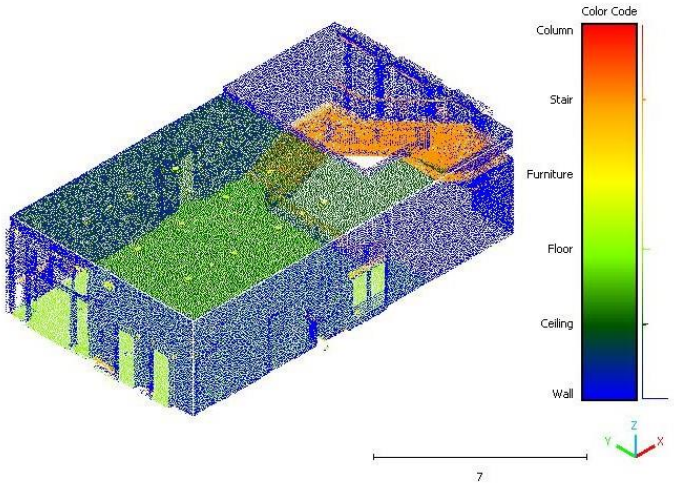
To classify the dataset TUM-F0, the classification model includes the datasets TUM-F1, TUM-F2 and two additional datasets, one representing a room and the second representing offices. These datasets provide information whether a certain geometry belongs to a specific class instance. To classify points that belong to staircase geometries, the classification model is trained on datasets including geometries representing staircases. Once the random forest classifier has established the model, its process is evaluated by publishing performance metrics. In the context of instances classified as wall instances, the precision, as illustrated in the accompanying Table 5 is returned as 0.94, while the recall is calculated as 0.97. Recall values reach their highest value for planar surfaces present in walls, ceilings, and floors, yet exhibit a decrease for instances categorized as furniture. This refers to the complexity within the designated neighboring radius used for computing geometric features. Specifically, points assigned to furniture instances show a dissimilarity in their geometric features due to the complexity of the surfaces containing furniture points within the specified radius. The increased complexity represents a challenge for the model, as points associated with these structures may share geometric characteristics with other labels rather than with the same label. For instance, the model might classify furniture points as Wall instances, particularly when they are situated in close proximity to points corresponding to walls. In contrast, planar surfaces exhibit more uniform geometric structures within the points neighborhood associated with the instance. However, an overall accuracy value of 0.95 indicates that the random forest classification model has precisely identified patterns and relationships in the data.



	Precision	Recall	f1-score	Support
11.0	0.94	0.97	0.95	79372
22.0	0.99	0.99	0.99	30456
33.0	0.98	0.99	0.99	22283
44.0	0.90	0.82	0.86	25605
55.0	0.97	0.88	0.92	7652

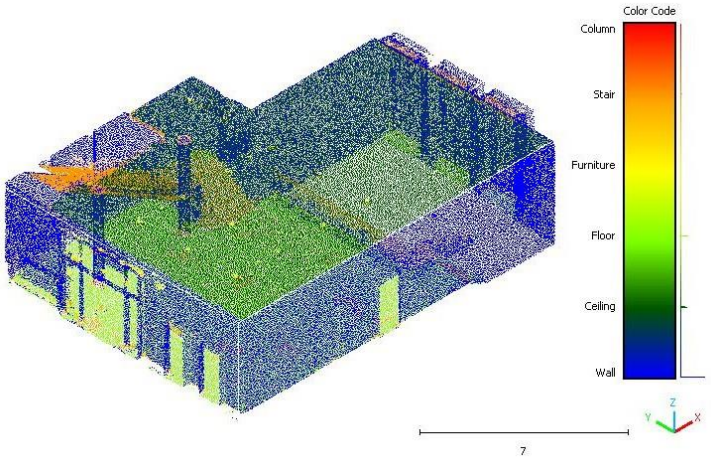
Table 5: Classification of dataset TUM-F0 and the performance metric of its classification model with an accuracy of 0.95.

The process is repeated for classifying both datasets, TUM-F1 and TUM-F2. For classifying TUM-F1, the model is trained on TUM-F0, TUM-F2, and two further datasets. For classifying TUM-F2, the model is trained on TUM-F0, TUM-F1, and three additional datasets. The third dataset that is used for training represents a hall with columns, because dataset TUM-F2 includes columns. To classify this dataset effectively, a model trained to learn patterns between geometric features and classes involving columns is necessary. Previous uses of TUM-F2 lacked an increased representation of the column class, resulting in low recall values. To address this, a dataset with increased column presence is employed. Despite columns being misclassified as walls, other objects are correctly assigned, as shown in Table 7. Without the additional dataset, the classifier mixes classes due to the underrepresentation of the column class.



	Precision	Recall	f1-score	Support
11.0	0.93	0.97	0.95	80919
22.0	0.99	0.99	0.99	33399
33.0	0.98	0.99	0.99	23188
44.0	0.90	0.82	0.86	25526
55.0	0.96	0.85	0.90	6368
66.0	0.81	0.54	0.65	471

Table 6: Classification of dataset TUM-F1 and the performance metric of its classification model with an overall accuracy of 0.95.



	Precision	Recall	f1-score	Support
11.0	0.92	0.95	0.93	117900
22.0	1.00	1.00	1.00	128593
33.0	0.98	0.98	0.98	133354
44.0	0.90	0.78	0.83	25555
55.0	0.97	0.87	0.92	7604
66.0	0.89	0.85	0.87	14083

Table 7: Classification of dataset TUM-F2 and the performance metric of its classification model with an overall accuracy of 0.96.

4.2 Experimental Result on the Staircase Parameter Extraction

Once the staircase has been labeled in the entire dataset, it is further processed to extract numerical values to be inserted into a parametric model, that defines the overall shape of the parametric model. This process involves extracting points labeled as stair, applying a grid to shift the entire staircase upwards, removing points associated with a railing by using DBSCAN and dividing the staircase into different parts based on the number of landings within the staircase. In the case of dataset TUM-F0 and TUM-F1 the staircase is divided into three runs and corresponding two landings. The first run aligns with an increasing x-axis, the second run aligns with a decreasing y-axis and the third run aligns with a decreasing x-axis. Dataset TUM-F2 contains two runs and corresponding one landing. The first run is aligned towards an increasing y-axis, the second run aligns with a decreasing x-axis.

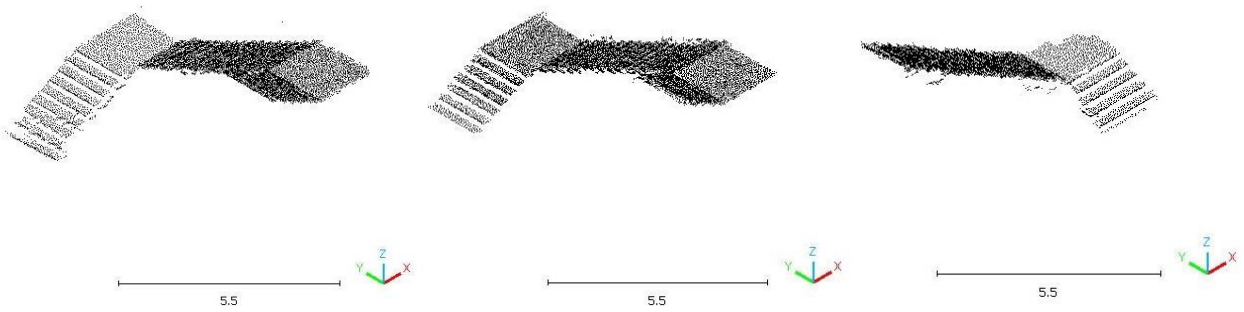


Figure 21: Classification of dataset TUM-F2.

4.3 Experimental Result on the Creation of Staircase Models

The division of the modified staircase prepares the point cloud for the extraction of numerical values, which describe the staircase's width, length, height, number of steps, tread height and tread width.

TUM-F0						
	Width	Length	Height	Number of Steps	Tread Height	Tread Width
Part a	1.91 m	2.78 m	1.38 m	9	0.14 m	0.31 m
Part b	1.86 m	4.82 m	2.48 m	16	0.15 m	0.30 m
Part c	1.90 m	1.49 m	0.76 m	5	0.13 m	0.30 m
TUM-F1						
Part a	1.75 m	2.14 m	1.03 m	7	0.13 m	0.30 m
Part b	1.95 m	4.65 m	2.38 m	16	0.14 m	0.29 m
Part c	1.86 m	1.19 m	0.56 m	4	0.11 m	0.30m
TUM-F2						
Part a	2.03 m	0.82 m	0.86 m	5	0.14 m	0.16 m
Part b	1.70 m	5.38 m	3.04 m	18	0.16 m	0.30 m

Table 8: Parametric values for datasets TUM-F1 and TUM-F2.

To establish the reference planes for aligning the staircase's height within the dataset TUM-F0, the extracted heights must be aggregated. With reference to the provided Table 8, the floor height is established at 4.63 meters. Moreover, the number of steps required to reach the desired height is specified. To determine the total number of steps, the counts for parts a, b, and c are summed, resulting in a total of 30 steps. This ensures the creation of 30 steps within the given height. The calculation involves dividing 4.63 meters by 30, yielding approximately 0.15 m per step.

Once establishing these parameters, the staircase is generated with a user-defined orientation, guided by visual indicators within the point cloud. This involves ensuring perpendicularity between the stair runs and aligning staircase parts a and c horizontally, while part b is aligned vertically in the xy plane. Modifying the direction automatically situates landing treads. The directional change automatically divides the staircase into three separate entities, each possessing modifiable dimensions in terms of width and height. While the width parameter is adjusted by inputting the corresponding value, the length of the staircase is contingent on the tread depth, a parameter defined within the instance properties of the system family stair, depicted in Figure 22. All parameters extracted from the point cloud data are incorporated into the introduced instance family parameters. The generated stair family along with the PCD is depicted in Figure 23.

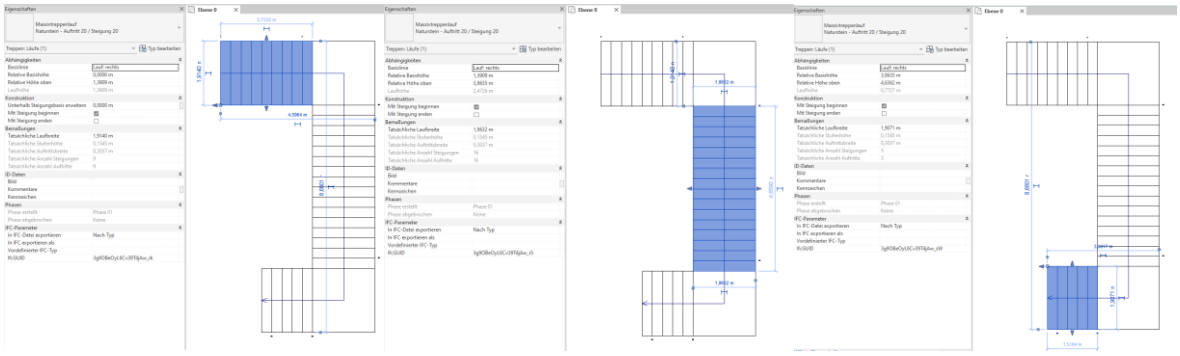


Figure 22: Dimensional Constraints within three parts in the entire staircase.

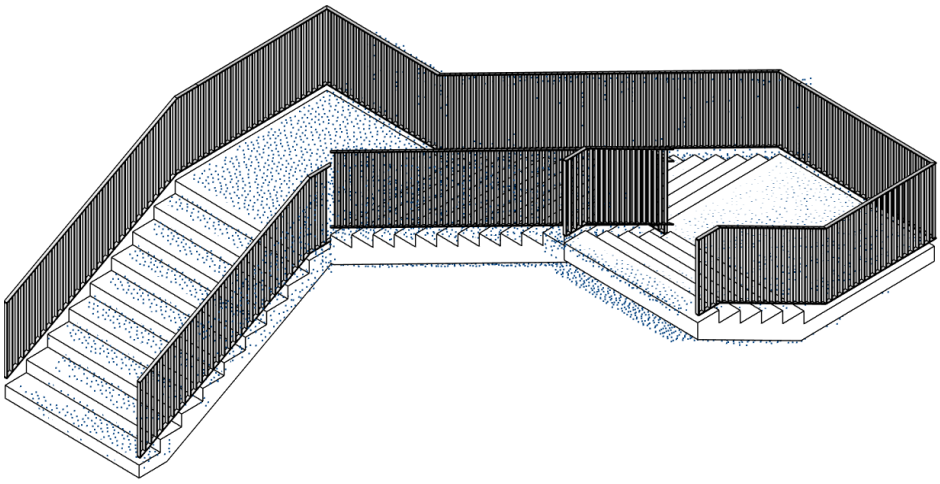


Figure 23: Dataset TUM-F0 represented with Point Cloud and Parametric Model in Autodesk Revit.

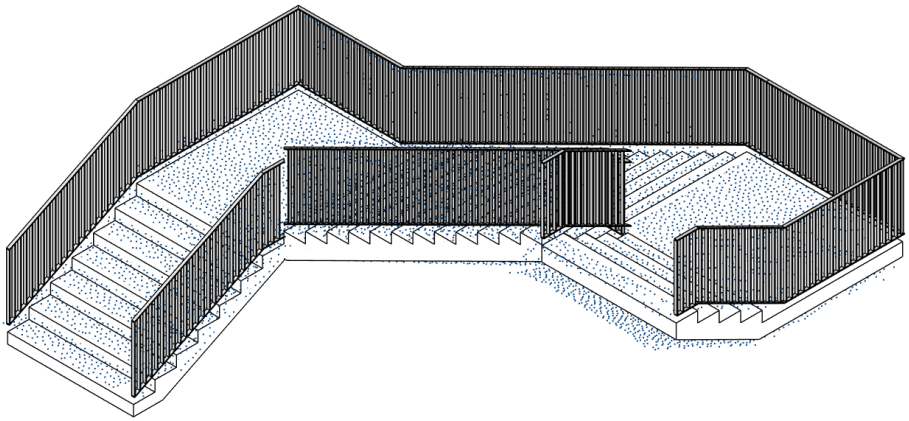


Figure 24: Dataset TUM-F1 represented with Point Cloud and Parametric Model in Autodesk Revit.

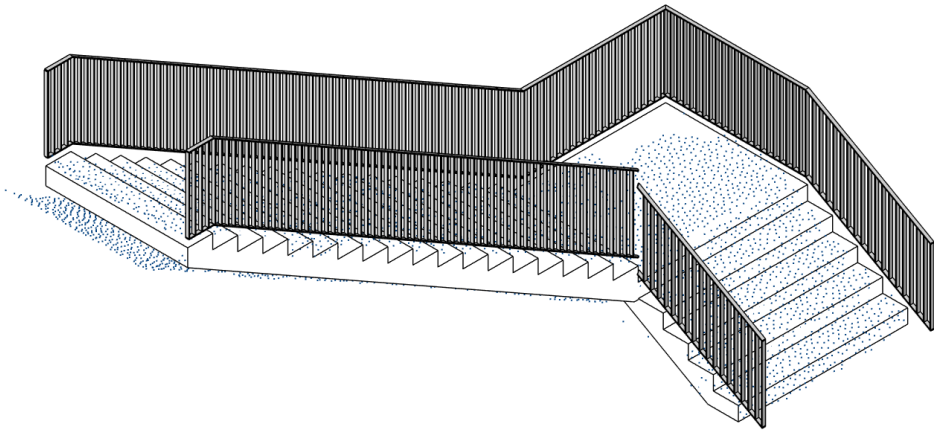


Figure 25: Dataset TUM-F2 represented with Point Cloud and Parametric Model in Autodesk Revit.

Autodesk Revit is extended through the visual programming interface Autodesk Dynamo. Dynamo enables users to create scripts, ensuring access and modification of specific parameters within families. This section details a workflow consisting of two primary steps:

1: Preprocessing: Numerical values of the point cloud are extracted by setting up python scripts. The extracted parametric values are then archived in a Microsoft Excel file.

2: Parametric Modeling: The Microsoft Excel file containing archived parametric values is imported into Autodesk Dynamo. Dynamo provides functionalities to read and process data from Excel, facilitating access to the stored parametric values. Using the imported parametric values, Dynamo scripts are created to parametrically define the geometry of staircases.

The workflow demonstrates the integration of Revit, Dynamo, and additional tools like Python scripting and Excel, offering a workflow for incorporating PCD and achieving automatic parametric modeling within the BIM authoring environment (Lee et al., 2023).

The provided script is utilized to extract numerical values as outlined in section 3.3.1. In addition to extracting values, the script computes the sum of steps and the sum of heights for each file type. Moreover, it calculates the average values for tread height and tread width, yielding mean values for these parameters. The script concludes by generating an Excel file that encapsulates the calculated numerical parameter values for the targeted dataset. The resulting tables below represent the extracted values for datasets TUM-F0, TUM-F1, and TUM-F2.

File Type	Steps	Width	Length	Height	Tread Height	Tread Width
a	9	1,9139509	2,7886763	1,3795312	0,137953123	0,309852918
b	16	1,8632183	4,8230004	2,4858317	0,146225393	0,301437526
c	5	1,9070921	1,4988875	0,7602551	0,126709176	0,299777508
Total/Average	30			4,625618	0,154187266	0,303689317

Table 9: Excel file containing parametric values within dataset TUM-F0.

File Type	Steps	Width	Length	Height	Tread Height	Tread Width
a	7	1,745069	2,1362343	1,0302789	0,128784859	0,305176327
b	16	1,9458756	4,6456127	2,3832373	0,140190431	0,290350795
c	4	1,8612299	1,1918678	0,5607626	0,112152518	0,297966955
Total/Average	27			3,9742788	0,14719551	0,297831359

Table 10: Excel file containing parametric values within dataset TUM-F1.

File Type	Steps	Width	Length	Height	Tread Height	Tread Width
a	5	2,029844	0,816208	0,8606868	0,1434478	0,1632416
b	18	1,701221	5,388573	3,0374308	0,159864778	0,299365167
Total/Average	23			3,8981176	0,169483373	0,231303383

Table 11: Excel file containing parametric values within dataset TUM-F2.

The process of defining the geometry of a staircase within Autodesk Revit and Autodesk Dynamo involves several steps. To begin, the corresponding system family stair must be chosen, and the total number of steps is specified using the "desired number of steps" parameter, as illustrated in Figure 26. With this parameter established, the actual staircase is drawn. The alignment, number of runs, landings, and steps in each run are determined through visual inspection. Taking the example of dataset TUM-F0, it's evident that the staircase comprises three runs and two landings. The first run aligns parallel to an increasing x-axis, the second run aligns parallel to a decreasing y-axis, and the third run aligns parallel to a decreasing x-axis. This visual information is then utilized to draw the staircase. After defining the runs and landings, the parametric values undergo refinement through the application of a Dynamo script. This script aids in further refining the specific characteristics and details of the staircase by importing the extracted parametric values.

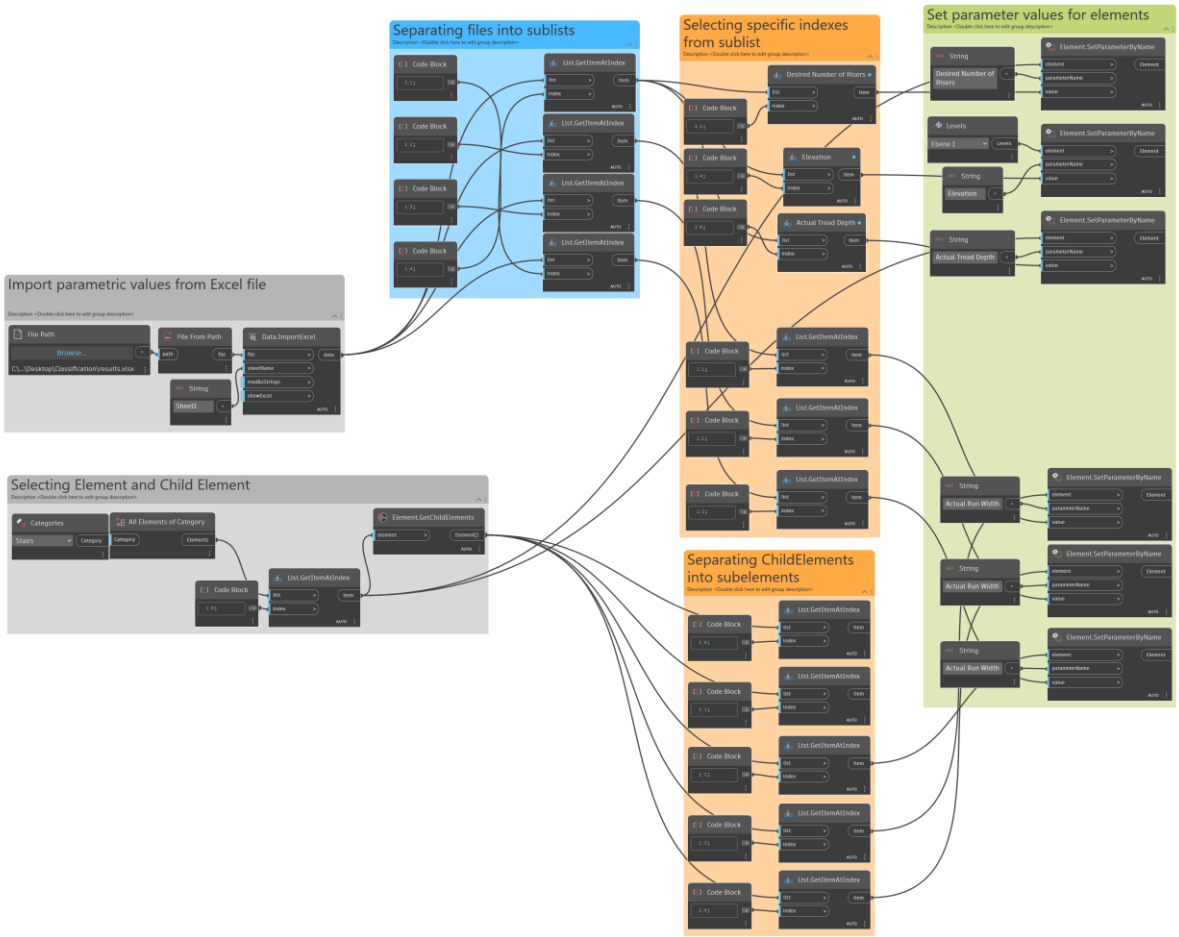


Figure 26: Dynamo Script for the automatic adjustment of the staircase parameters.

The script initiates by importing an Excel file, leveraging the "file path" node to select the Excel file's location. After specifying the sheet name, relevant information is extracted from the file, presented as four lists corresponding to four columns. These lists are examined individually in the subsequent step to capture specific parameters of interest. In the context of the TUM-F0 dataset, the focus is on values such as the sum of all heights, the sum of steps, average tread width, and the actual run width for each of the three runs within the stair geometry. The node "List.GetItemAtIndex" is employed within the "Selecting specific indexes from sublist" group to access these values. Given that the system family stair involves nested families such as runs and landings, the script utilizes the "Element.GetChildElement" node. This ensures access to the parameters of each run and landing within the staircase. A crucial step involves adjusting the run width, which is achieved through the extraction of width values for each run defined by the "Width" parameter. The node group "Separating ChildElements into subelements" facilitates the individual access to each run. The width value for each run is then linked to the corresponding value extracted from the Excel file. Through the interconnection of nodes, the script automates the parametric modeling of the staircase. In contrast to manually setting values through instance properties, the script ensures the automatic adoption of values calculated by the Python script. The visual outcomes of the parametric staircase closely resemble those achieved through the previous modeling process. However, the efficiency gains are significant, leading to a reduction in modeling time.

4.4 Evaluation

To assess the parametric values derived from the point cloud, a comparative analysis with measurements obtained through an inch rule is processed. By analyzing the disparities between these two sets of values, an evaluation of the methodology's accuracy is processed. The closer the correspondence between these values, the higher the accuracy attributed to the developed methodology. This process allows for a statement on the reliability and precision of the applied methodology in capturing and representing the complexities of the staircase geometry. The assessment is conducted based on the three datasets, comparing, and analyzing the results to determine the consistency and effectiveness of the methodology across different sets of data.

	Part	TUM-F0	TUM-F1	TUM-F2
Width	a	0.05 m	0.1 m	0.03 m
	b	0.06 m	0.01 m	0.03m
	c	0.04 m	0.01 m	-
Tread Height	a	0.01 m	0.01 m	0.02 m
	b	0.00 m	0.00 m	0.00 m
	c	0.02 m	0.03 m	-
Tread Width	a	0.01 m	0.00 m	0.13 m
	b	0.00 m	0.01 m	0.00 m
	c	0.00 m	0.00 m	-

Table 12: Deviation between actual and extracted parameter values.

The accuracy of the methodology is reflected by comparing the calculated and measured values with each other. Notably, tread height values demonstrate the highest accuracy, with a deviation of 0-3cm. Following closely are the width values, displaying a 0-6cm deviation. Width values tend to be larger in calculations. The calculation method involves identifying the minimal and maximal x or y coordinates within the point cloud of each run to determine the staircase's width. As these points are present in the landing, the width of the staircase run is influenced, resulting in larger values. The least accurate results are observed for tread width, with a deviation ranging from 0-13cm. This discrepancy arises from the calculation of tread depth, which is derived by dividing the run's length by the corresponding number of steps within the run. The run's length is determined by identifying the minimum and maximum values of y or x coordinates within the point cloud. The calculated width is then subtracted from the total length to exclude the landings' length from the point cloud. This subtraction, if a larger value, leads to an inaccurate dimensional length and, consequently, an imprecise tread depth.

5 Discussion

Section 1 introduced the topic of the thesis, followed by its motivation and objective. In brief, the research on DT has observed an increase in recent decades, particularly as the concept, originating in the manufacturing industry, transitions into the construction industry. However, given its novelty in this industry, the process is acknowledged to be prone to errors and labor-intensive. The thesis aims to address this challenge by formulating an automated methodology for creating a DT from PCD, with a specific focus on staircases. DT represents the status of the construction site with PCD serving as a surface representation of the object. Since PCD contains geometric information like coordinates, colors, and surface normals, PCD contributes to the recreation of a DT by applying AI methods, that process the PCD to return geometric information.

Section 2 reviewed literature, clarifying concepts essential for the subsequent section. This involved an exploration of themes such as digital twining, scan-to-BIM, PCD and geometric features. Furthermore, the section delved into the details of point cloud semantic enrichment, demonstrating the application of classification and clustering algorithms to PCD. Preceding the conclusion of the section with a discussion on research gaps, detailed introductions were made to concepts of geometric model reconstruction, including implicit and explicit representation, along with parametric modeling.

Section 3 illustrated the steps involved in classification, clustering, parameter extraction, and geometric model reconstruction. This included interpreting performance metrics, analyzing appropriate parameter values for clustering, and establishing rules for extracting necessary dimensions. The section also explained the generation of parametric staircase structures, exploring various combinations and orientations of landings and runs. A second approach was introduced, utilizing pre-established parametric staircase models in Autodesk Revit.

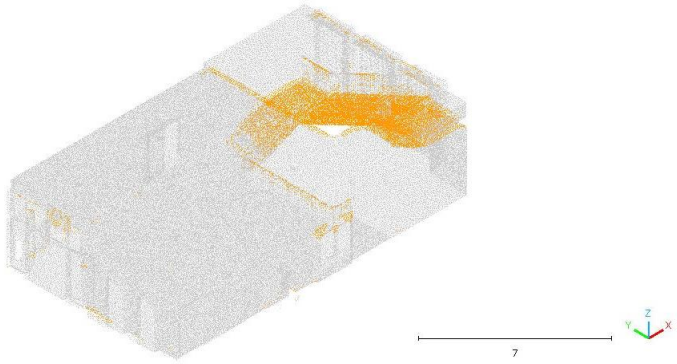
Thus, the section explored two methodologies for parametric modeling – one by devising a written script and another by using built-in object families in Autodesk Revit.

In section 4 the established methodology is implemented on three different datasets. The application underscores the adaptability of the methodology to diverse scenarios. By creating a Dynamo script, a direct link between a Python script that extracts parametric values, and the Revit user interface is established, facilitated through an Excel file, containing relevant parameters for the parametric modeling of the staircase. However, the adjustment of the script becomes necessary based on observations of the geometric characteristics of the staircase. Subsequently, each script within the process is executed, yielding the essential information for the reconstruction of the geometric model. This streamline approach enhances the efficiency of the DT process. The section concludes with an evaluation, wherein calculated values are compared with measured values using an inch rule. This comparative analysis serves to validate the accuracy and reliability of the reconstructed geometric model.

5.1 Comparison between classification and clustering methods

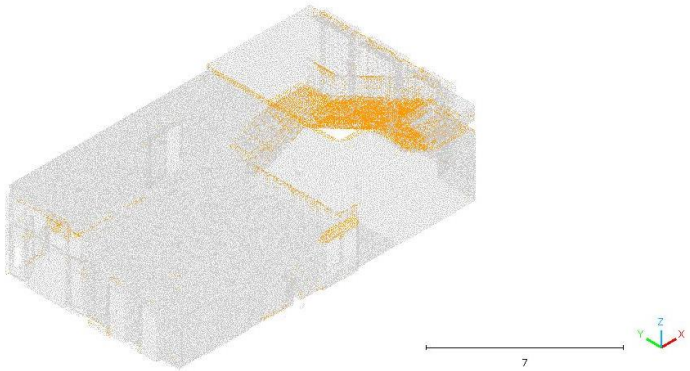
While both the random forest classifier and decision tree classifier are categorized as machine learning algorithms, notable distinctions arise between these two classification models. A random forest classification model comprises an ensemble of decision trees, each trained on a random subset of the data. The final prediction is determined through a voting mechanism across all the individual trees. On the contrary, a decision tree classification model consists of a singular tree structure, that learns patterns based on the features of the input data. The susceptibility to overfitting is a characteristic of decision tree classification models due to their lack of variance. Overfitting occurs when the model memorizes the training data rather than capturing its underlying patterns. To address this, random forest classification models are employed, incorporating predictions from multiple trees to enhance robustness against overfitting. The increased diversity among the trees contributes to this resilience. However, it is important to note that the time required for both prediction and training is higher for random forest classification models compared to decision tree classification models. This is referred to the augmented number of trees in the data. To provide a practical comparison, the classification prediction for the TUM-F1 dataset involves training both a random forest

classification model and a decision tree classification model. The results of this comparison in the classification, along with the associated performance metrics, are illustrated in Figure 27 and Figure 28.



	Precision	Recall	f1-score	Support
11.0	0.92	0.98	0.95	53329
22.0	0.99	0.99	0.99	23387
33.0	0.98	0.99	0.99	22950
44.0	0.88	0.61	0.72	9344
55.0	0.95	0.84	0.89	7210
66.0	0.98	0.86	0.92	447

Figure 27: Random Forest Classification Model Performance Metric TUM-F1.



	Precision	Recall	f1-score	Support
11.0	0.93	0.93	0.93	55011
22.0	0.98	0.98	0.98	26235
33.0	0.98	0.98	0.98	23703
44.0	0.65	0.65	0.65	9293
55.0	0.84	0.84	0.84	7237
66.0	0.85	0.87	0.86	439

Figure 28: Decision Tree Classification Model Performance Metric TUM-F1.

As depicted, the utilization of the random forest classification model results in an improved accuracy when predicting points associated with the staircase instance. For instance, as shown in Figure 28, the second landing of the staircase structure was not initially included in the staircase instance due to its similarity to floors and ceilings. The use of the random forest classification addresses this issue, as illustrated in Figure 27.

This improvement can be attributed to the model's capability of handling complex relationships within the data. The random forest classification model allows each tree to concentrate on specific aspects of the data and subsequently combining these diverse trees within the ensemble, resulting in an enhancement in prediction accuracy and robustness.

The preceding content presented a comparison between classification models. In the following, a comparison between clustering algorithms, specifically DBSCAN and region-growing, which have been employed to group similar data points is provided. While both algorithms partition the data into groups, their fundamental difference lies in their approach to achieve this target. DBSCAN groups points based on their density reachability to each other. Density reachability is determined by a neighborhood within a specified radius, encompassing a specific number of points set by the parameter's eps and minPts. This algorithm is particularly effective in eliminating outliers that deviate from the PCD structure. In the context of the thesis, DBSCAN has primarily been utilized for outlier removal and the detection of specific structures within PCD, such as the handrail.

A simplified version of the region-growing algorithm has been applied to count the number of steps within the staircase. Each step is defined by a plane containing points with varying x and y coordinates but nearly identical z coordinates. The algorithm uses the similarity in z coordinates to group points into clusters based on a threshold that determines how much the z values may differ. Subsequently, the number of clusters is counted to determine the number of steps within the run.

DBSCAN focuses on the quantity of points within the point cloud, whereas region-growing algorithm expands regions based on similar values within the point's geometric features. Consequently, the region-growing algorithm is applied for various clustering problems, given the abundance of geometric features derived from the point cloud.

5.2 Contributions and Limitations

This section introduces an evaluation of the contributions and limitations presented in the thesis. The significance of understanding the impact and strength of the work undertaken is emphasized by revealing the advancements made in the field. Simultaneously, the acknowledgment of limitations provides transparency and sets the context for future research. Notable contributions and potential constraints inherent in the findings and methodologies of the thesis will be provided.

The thesis has devised a methodology aimed at streamlining the DT creation process from PCD. This has been achieved by illustrating approaches for extracting geometric information that serve as the foundation for parametric modeling. Additionally, the thesis explored and applied classification and clustering methods to fulfill its objective. However, employing these classification and clustering methods demands an increased amount of annotated data and increased computing power for model training. The proposed solution primarily addresses a low level of detail (LOD), focusing on essential parameters such as width, length, height, tread height and tread width to define the staircase. This choice results in a decreased information of PCD during its processing. Notably, aspects like handrailing height, specific landing measurements and staircase thickness have not been considered. For instance, a more detailed consideration of landing measurements could have led to more accurate results for the length parameter. While the developed methodology has proven effective in streamlining DT creation, there are inherent limitations related to the LOD and the requirements for model training.

6 Conclusion

The proposed methodology utilizes PCD to generate a DT of staircase geometries. The outcome of this approach comprises a geometric parametric model that aligns with the structure of the PCD. The thesis introduces a systematic pipeline for creating a DT from PCD. When applied to additional datasets, this pipeline serves to reduce labor and minimize information loss throughout the DT creation process. Initiating with the segmentation of PCD components, the approach focuses on the staircase geometry in subsequent steps. The PCD representing the staircase undergoes further processing to extract dimensional values. The proposed methodology offers several benefits. It enables the direct extraction of parameters from the file containing XYZ coordinates. The dimensioning of the PCD is addressed through clustering, classification and algorithms that are linked with the geometric file structure. Upon establishing classification and clustering algorithms specifically for staircase structures, these will be applied to diverse datasets to extract parametric values. The methodology not only accelerates the parameter extraction process but also enhances precision due to its direct correlation with the coordinates of the file of the PCD. The methodology concludes by incorporating these extracted values into the parametric model, yielding a DT of staircase structures, which ensures consistency between the DT and the PCD.

References

Abreu, N., Pinto, A., Matos, A., & Pires, M. (2023). Procedural point cloud modelling in scan-to-BIM and scan-vs-BIM applications: a review. *ISPRS International Journal of Geo-Information, 12(7)*, 260.

Awwad, T., Zhu, Q., Du, Z., & Zhang, Y. (2010). An improved segmentation approach for planar surfaces from unstructured 3D point clouds. *The Photogrammetric Record, 25(129)*.

Bello, S., Yu, S., Wang, C., Adam, J., & Li, J. (2020). Review: Deep learning on 3D point clouds. *Remote Sensing, 12(11)*, 1729.

Borrmann, A., König, M., Koch, C., & Beetz, J. (2015). *Building Information Modeling: Technologische Grundlagen und industrielle Praxis.* Springer-Verlag.

Breimann, L. (2001). Random Forests. *Machine Learning, 45*, 5-32.

Daskalova, M. (2021). *The 'digital twin'–a bridge between the physical and the digital world.* Retrieved from Cobuilder: https://cobuilder. com/en/the-digital-twin-a-bridge-between-the-physical-and-the-digital-world/

Duell, R., Hathorn, T., & Hathorn, T. (2013). *Autodesk Revit Architecture 2014 Essentials: Autodesk Official Press.* John Wiley & Sons.

Eastman, C. (2011). *BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors.* John Wiley & Sons.

Fayez, T.-K., Landes, T., & Grussenmeyer, P. (2007). Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007. Vol. 36*.

Gan, G., Ma, C., & Wu, J. (2020). *Data Clustering: Theory, Algorithms, and Applications.* Society for Industrial and Applied Mathematics.

Hackel, T., Schindler, K., & Wegner, J. (2016). Contour Detection in Unstructured 3D Point Clouds. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 1610-1618.

Hahsler, M., Piekenbrock, M., & Doran, D. (2019). dbscan: Fast density-based clustering with R. *Journal of Statistical Software, 91*, 1-30.

He, P., Ma, Z., Fei, M., Liu, W., Guo, G., & Wang, M. (2022). A Multiscale Multi-Feature Deep Learning Model for Airborne Point-Cloud Semantic Segmentation. *Applied Sciences, 12(22)*, 11801.

Hegelich, S. (2016). Decision trees and random forests: Machine learning techniques to classify rare events. *European Policy Analysis, 2(1)*, 98-120.

Jesse, D., & Goadrich, M. (2006). The relationship between Precision-Recall and ROC curves. *Proceedings of the 23rd international conference on Machine learning*, pp. 233-240.

Khaloo, A., & Lattanzi, D. (2017). Robust normal estimation and region growing segmentation of infrastructure 3D point cloud models. *Advanced Engineering Informatics*, 1-16.

Kritzinger, W., Karner, M., Traar, G., Henjes, J., & Sihn, W. (2018). Digital Twin in manufacturing: A categorical literature review and classification. *Ifac-PapersOnline, 51(11)*, 1016-1022.

Lee, Y.-C., Leite, F., & Ma, J. (2023). A parametric approach towards semi-automated 3D as-built modeling. *Journal of Information Technology in Construction*.

Liu, S., Zhang, M., Kadam, P., & Jay Kuo, C.-C. (2021). *3D Point Cloud Analysis: Traditional, Deep Learning, and Explainable Machine Learning Methods.* Springer International Publishing.

Lu, B., & Yang, W. (2019). Matching algorithm of 3D point clouds based on multiscale features and covariance matrix descriptors. *IEEE Access*.

Lu, R., & Brilakis, I. (2019). Generating bridge geometric digital twins from point clouds. *EC3 Conference 2019, Vol. 1*, (pp. 367 - 376). University College Dublin.

Mehranfar, M., Braun, A., & Borrmann, A. (2022). A hybrid top-down, bottom-up approach for 3D space parsing using dense RGB point clouds. *ECPPM 2022-eWork and eBusiness in Architecture, Engineering and Construction*.

Nasteski, V. (2017). An overview of the supervised machine learning methods. *Horizons. b*.

Neufert, E., & Neufert, P. (2012). *Architects' data.* John Wiley & Sons.

Olsen, M., Kuester, F., & Chang, B. (2010). Terrestrial Laser Scanning-Based Structural Damage Assessment. *Journal of computing in civil engineering, Vol.24 (3)*, 264-272.

Opoku, D., Perera, S., Osei-Kyei, R., & Rashidi, M. (2021). Digital twin application in the construction industry: A literature review. *Journal of Building Engineering, 40*, 102726.

Pan, Y. (2023). *Creating an information-rich digital twin of indoor environments by interpretation and fusion of image and point-cloud data (Doctoral dissertation).* Technische Universität München.

Rocha, G., & Mateus, L. (2021). A survey of scan-to-BIM practices in the AEC industry-a quantitative analysis. *ISPRS International Journal of Geo-Information, 10(8)*.

Shah, J., & Martti, M. (1995). *Parametric and feature-based CAD/CAM: concepts, techniques, and applications.* John Wiley & Sons.

Sharma, H., & Kumar, S. (2016). A survey on decision tree algorithms of classification in data mining. *International Journal of Science and Research (IJSR), 5(4)*, 2094-2097.

Son, H., Kim, C., & Turkan, Y. (2015). Scan-to-BIM-an overview of the current state of the art and a look ahead. *ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction (Vol. 32)* (p. 1). IAARC Publications.

Tang, P., Huber, D., Akinci, B., Lipman, R., & Lytle, A. (2010). Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques. *Automation in construction, 19(7)*, 829-843.

Tóvári, D., & Pfeifer, N. (2005). Segmentation based robust interpolation-a new approach to laser data filtering. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*.

Voelcker, H., & Requicha, A. (1977). *Constructive Solid Geometry.*

Declaration

I hereby declare that I wrote the present thesis on my own. Only the sources and resources expressly named in the work were used. Literally or analogously adopted ideas I have identified as such.

I also assure that the present work has not yet been used as the basis for another examination procedure.

Munich, 15.03.2024

Alexandra Bulla

Alexandra Bulla

[redacted]