Advanced Engineering Informatics 51 (2022) 101497

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Full length article

Ensemble-learning approach for the classification of Levels Of Geometry (LOG) of building elements

Jimmy Abualdenien [*], André Borrmann

Chair of Computational Modeling and Simulation, Technical University of Munich, Arcistrasse 21, 80333 Munich, Germany

ARTICLE INFO

Keywords:
Building Information Modeling (BIM)
Level Of Development (LOD)
Level Of Geometry (LOG)
Geometric Complexity
Ensemble models

ABSTRACT

The provision of geometric and semantic information is among the most fundamental tasks in BIM-based building design. As the design is constantly developing along with the design phases, there is a need for a formalism to define its maturity and detailing. In practice, the concept of Level of Development (LOD) is used to specify what information must be available at which time. Such information is contractually binding and crucial for different kinds of evaluations. Numerous commercial and open-source BIM tools currently support the automatic validation of semantic information. However, the automatic validation of the modeled geometry for fulfilling the expected detailing requirements is a complex and still unsolved task. In current practice, domain experts evaluate the models manually based on their experience. Hence, this paper presents a framework for formally analyzing and automatically checking the Level Of Geometry (LOG) of building information models. The proposed framework first focuses on generating a LOG dataset according to the popular LOD specifications. Afterwards, multiple geometric features representing the elements' complexity are extracted. Finally, two tree-based ensemble models are trained on the extracted features and compared according to their accuracy in classifying building elements with the correct LOG. Measuring the modeling time showed a 1.88–2.80-fold increase between subsequent LOGs, with an 8–15-fold increase for LOG 400 compared to LOG 200. The results of classifying the LOG indicated that the combination of 16 features can represent the LOG complexity. They also indicated that the trained ensemble models are capable of classifying building elements with an accuracy between 83% and 85%.

1. Introduction

The design and detailing decisions made throughout the building design phases significantly influence a project's time, effort, and cost [1,2]. Starting from the schematic design, the project size, building shape, and materiality are defined broadly in order to explore the different possible options. The decisions made in the early phases form the design intent, representing the basis for further detailing [1,3]. Such detailing includes refining the elements' geometry and evaluating the different combinations of material layers.

As construction projects are multi-disciplinary, a fundamental pillar for integrating building information models is describing the required elements' maturity at every milestone and for every deliverable through the design phases. This is crucial for the overall collaboration among the project participants because it acts as an agreement on (what) information should be available at what time (when). Based on the available information, it can be decided what the model can be used for (purpose), which makes it possible to determine what model deliverables are expected from the actors involved (who) [4]. The exchange of complete and compliant Building Information Modeling (BIM) data within the Architecture, Engineering, and Construction (AEC) industry is crucial, as it is prescribed in legal agreements, where the content of the individual elements is specified. Accordingly, a common legal framework for organizing this data is required.

Data quality is described by compliance with its requirements' characteristics [5]. More specifically, the quality of building information is expressed by the correctness and completeness of the topological relationships, geometric detailing, and semantics. Various guidelines have been published to deliver a standard that practitioners can use as a basis for a common language in their projects. When describing the detailing decisions, the Level of Development (LOD) [6], is a popular concept for defining the content of a model at a certain point during the design process. The LOD refers to the completeness and reliability of the building elements' information.

For more than a decade, practitioners have relied on the LOD terminology to specify which information they need to carry out and deliver their tasks [2,7,8]. However, as the different LOD definitions are

[*] Corresponding author.
E-mail address: jimmy.abualdenien@tum.de (J. Abualdenien).

https://doi.org/10.1016/j.aei.2021.101497
Received 8 September 2021; Received in revised form 25 November 2021; Accepted 6 December 2021
Available online 10 January 2022

loosely defined [8,9], each practitioner has a different interpretation of what a specific LOD means and which information should be present in the model [7–9]. Such inconsistencies cause severe miscommunication and additional expenditure, which increase project risks [2,7]. Therefore, multiple efforts have been dedicated to developing comprehensive specifications worldwide to provide a consensus on the required information at the different LODs (more details are provided in Section 2.3). The most popular among these are the BIMForum's LOD specification [6] and Trimble's Project Progression Planning [10], which are used as the basis for the research conducted in this paper.

The geometric detailing of building elements is essential for carrying out different kinds of analyzes and evaluations (e.g., energy analysis, evaluation of design options, and cost estimation) that support decisions during design and construction [7,11–13]. For example, according to the BIMForum's specification, performing clash detection or analyzing the constructability of the outer building shell requires modeling the precise geometry (available from LOD 300) and connections between the elements (available from LOD 350). The requirements of each LOD comprise both semantic information (a.k.a. Level of Information (LOI)) and geometric detailing (a.k.a. Level Of Geometry (LOG)). The LOI is represented by a set of properties, whereas the LOG is described by the geometric parts that need to be modeled, like modeling the overall shape precisely or the necessary reinforcement parts.

As the LODs are referenced in contracts and BIM execution plans, their requirements must be fulfilled when delivering and exchanging building models. In this regard, automatically checking the completeness of the semantic information is straightforward [14] and supported by numerous commercial and open BIM tools. However, automatically checking that the detailing of the modeled geometry fulfills the expected LOG requirements is a complex and still unsolved task; currently, domain experts evaluate the models manually based on their experience. Therefore, the primary focus of this paper is to formally define the LOGs to identify a given BIM element's LOG in accordance with these specifications.

Machine Learning (ML) algorithms, including Support Vector Machines (SVM), Random Forests (RF), and Artificial Neural Networks (ANN), have demonstrated high performance in addressing non-linear multi-class classification and regression problems in different domains [15]. Generally, these approaches utilize statistics in order to extract generalizable and predictable patterns from a training dataset. Accordingly, the basic concept lies in implicitly deducing correlations between the provided data (input) and the expected result (output). In the AEC industry, the application of ML algorithms has become popular for multiple use-cases. For example, Zhang et al. developed a RF model for predicting the uniaxial compressive strength of lightweight self-compacting concrete [16]. Dong et al. trained an eXtreme Gradient Boosting (XGBoost) model [17] to predict concrete electrical resistivity for structural health monitoring [18]. Finally, Braun et al. developed a deep-learning model for supporting progress monitoring through detecting elements in point cloud data and comparing them to the BIM model [19].

This paper addresses the currently existing gap of determining the LOG of building elements by investigating the major characteristics representing the degree of detailing based on a formal metric. To this end, a set of different BIM element types (a.k.a. families) are modeled at multiple LOGs, and the geometric information of each level is investigated. In more detail, the elements are modeled using Autodesk Revit[1] and exported into triangulated meshes. Then, for each LOG, the geometric features are extracted, and their complexity is measured using a combination of multiple advanced geometry processing algorithms. Finally, RF and XGBoost models are trained on the extracted geometric features to classify the LOG of any given building element automatically. The contributions of this study are threefold: First, evaluating and measuring the necessary time and effort in modeling according to the common LOD specifications. Second, identifying the geometric features that are capable of representing the building elements' complexity across the LOGs. Third, evaluating the performance of state-of-the-art ensemble-learning models for classifying the LOG of elements. In this paper, *Shape Complexity* is a high-level term used to describe the overall shape composition, including the modeled parts on the different LOGs. Additionally, the term *Geometric Complexity* describes the geometric features necessary for representing the different shape parts, including vertices, edges, etc.

The paper is organized as follows : Section 2 discusses the background and related work, including shape complexity, LODs, and ensemble-learning. Section 3 provides an overview of the framework developed in this paper, explaining the process followed in modeling the different families to generate the LOG dataset. Additionally, Section 3 presents the geometric features selected to represent the building elements' complexity at the different LOGs. For classifying the LOG of building elements, RF and XGBoost models are developed and evaluated in Section 4. Moreover, Section 4 assesses the trained models' robustness by evaluating their performance on a re-meshed test dataset. Section 5 emphasizes the applicability of using the developed approach in practice via a real-world case study. Finally, Section 6 summarizes our results and presents an outlook for future research.

2. Background and related work

2.1. 3D shapes

The 3D representation of objects is a fundamental perspective for numerous domains, from computer graphics to BIM. Especially in BIM, the 3D representation of building elements is the primary way of defining the shape of a building and its components. It is also a fundamental aspect for performing a variety of tasks, including clash detection, quantity take-off, or even exploring the reliability of the building information across the design phases [11,20]. In BIM models, the 3D geometry is typically represented through two main approaches [21], (1) explicit modeling (a.k.a. boundary representation), which describes the geometrical surface characteristics, volume, and topology through a graph of faces, edges, and vertices, and (2) implicit modeling, which describes the geometric features through a sequence of operations that form the final representation when performed in the defined order.

A popular approach of explicit modeling is the polygon mesh representation [22–24]. Polygonal meshes require only a small number of polygons to represent simple shapes (regardless of their size). Additionally, a polygonal mesh has the necessary capability to comprehensively represent complex shapes with high resolution, capturing the salient surface features. Accordingly, simple shapes are represented by a few large polygons, while detailed and complex shapes are represented by many small polygons. Polygons comprise a set of vertices, which are interpolated through a connectivity graph to approximate the desired surface. On the other hand, Constructive Solid Geometry (CSG), extrusions, and sweeps are common operations of procedural modeling. In comparison to explicit modeling, procedural approaches have the advantage that the modeling history can be transported, which provides the potential for modifying the geometry in the receiving application. As however, misinterpretations are more likely when processing procedural descriptions, boundary representations are often favored over procedural representation in many BIM exchange scenarios [21].

Extracting the geometric features from building models is a fundamental part of the methodology presented in this paper. Hence, it is crucial to choose and follow a unified approach during the study. Since the implicit modeling approaches can also be represented using boundary representations, all the 3D shapes investigated in this paper were represented as polygon meshes.

[1] https://www.autodesk.com/products/revit/overview.

2.2. Shape complexity

The meaning and measurement of shape complexity varies according to different aspects. Processing geometric models can be as simple as iterating over a mesh's vertices, faces, and edges, or as complex as performing different calculations to extract information about curvature or shape topology. Numerous researchers have developed algorithms to retrieve the most dominant features of the different shapes [22], including detecting sharp edges, deducing surface patches, and decomposing the shape into smaller and meaningful shapes, a.k.a. segmentation [25]. Dominant features provide an essential description of the geometrical objects' resolution and detail. In the same context, Hanocka et al. and Nikhila et al. have developed MeshCNN [26] and PyTorch3D [27] by employing deep-learning approaches to analyze, process, and extract features from 3D shapes.

A popular classification for shape complexity was first introduced by Forrest, who defines three main types [28]: (1) geometric, which describes the shapes' basic features, such as lines, curves, faces, etc., (2) combinatorial, which refers to the topology of the shape, i.e., the number of components that it comprises, and (3) dimensional, which classifies the shape as 2D, 2.5D, or 3D. Other researchers have interpreted 3D shapes and their complexity through shape grammars [29]. Shape grammars describe the shape decomposition as a set of rules and a series of transformations, including addition, subtraction, rotation, etc.

Accordingly, defining what shape complexity means in the AEC industry requires the specification of which geometric features are essential for capturing the degree of maturity of building elements at the different LOGs [30].

2.3. Level of Development (LOD)

As a response to the need to have a consensus about what information should exist during the design process of building elements, various guidelines were published to deliver a standard that practitioners can use as a basis for a common language in their projects. Prior to the LOD concept, a relatively similar concept, a.k.a. Level of Detail (LoD), was already common in computer graphics. The LoD is used to bridge the graphical complexity and rendering performance of a computer program by regulating the amount of detail used to represent the virtual world. In computer graphics, the LoD concept is mainly concerned with geometrical detailing [31]. In the context of the data exchange standard CityGML, the LoD represents different levels of geometric and semantic complexity of a city model [32]. The software vendor VicoSoftware [10,33] was the first to apply the concept in a similar fashion to BIM models.

In the AEC industry, the term Level of Development (LOD) was favored over Level of Detail (LoD) as it represents the maturity, completeness, and reliability of the geometrical and semantical information provided by building elements [6]. The LoD concept has then been adopted and refined by the American Institute of Architects (AIA) to become LOD [34]. The AIA introduced a LOD definition that comprises five levels, starting from LOD 100 and reaching LOD 500. The BIMForum working group developed a new level, LOD 350, and published the Level of Development Specification based on the AIA definitions [6]. At the same time, Trimble's Project Progression Planning [10] was published and is widely used in practice.

Numerous countries, especially in Europe, have proposed different terms for their regions. In the UK, the Level of Definition [35] has been introduced. It consists of seven levels and introduces two components: Levels of model detail, which represents the graphical content of the models, and Levels of model information, which represents the semantic information. The Danish definition includes seven Information Levels that correspond roughly to the traditional project life-cycle stages [8]. Similarly, in Germany, the Modelldetaillierungsgrad (MDG) comprises 10 levels (010, 100, 200, 210, 300, 310, 320, 400, 510, 600) that also correspond to the project life-cycle stages [36]. The Italian LOD definition adopts the BIMForm's specification while adjusting it to seven levels with letters in ascending order from LOD A – LOD G [37]. In Switzerland, the LOD concept is based on the BIMForum's definitions, but at the same time, its usage is assigned to project life-cycle stages [38].

Recently, a similar concept was introduced by the European Standardization Organization (CEN) [39], which defines the term Level of Information Needs comprising specifications for LOG and LOI for supporting a particular use-case.

2.4. Supporting the design process using LODs

As the LODs provide means for specifying and communicating which information is expected to be present at a specific time, they were used by numerous practitioners and researchers for defining the required information throughout the design phases [14,40–42]. Abualdenien and Borrmann developed a meta-model approach for specifying the design requirements of individual families using the LODs, incorporating the information uncertainty [14]. In the same context Gigante-Barrera et al. included the LODs as an indicator for the necessary information within Information Delivery Manuals (IDMs). Abou-Ibrahim and Hamzeh developed a framework for applying lean design principles based on LODs [43]. Additionally, Grytting et al. introduced a conceptual model of a LOD decision plan, based on a set of interviews and use-cases, to support design decisions [44].

To support the decision-making process from the early design phases, Abualdenien et al. used the LODs to integrate the design process with energy simulations and structural analysis [45]. Additionally, Exner et al. proposed a LOD-based framework for comparing the different design variants and their detailing [46]. To exchange design requests and issues between projects participants, Zahedi et al. proposed a communication protocol that leverages the LODs to describe design requirements [13]. Finally, Abualdenien and Borrmann developed multiple visualization techniques to depict the information uncertainty associated with the LODs throughout the design phases [11].

2.5. Analysis and validation of LOGs

The process of adopting a LOD specification in a particular country (or even internally in individual firms) requires a comprehensive analysis and understanding of which geometric and semantic information should be present at each LOD. However, practitioners have an inconsistent understanding of the information necessary at each LOD [8,14]. The main reason is that although the specification of semantics is usually simplified to a list of properties, systematically checking the geometric detailing is an unresolved task.

In this regard, Leite et al. evaluated the modeling effort associated with generating BIM models at different LoDs. The authors have shown the need for an increased modeling time, ranging from doubling the modeling effort to eleven folding it, to detail models further to reach a higher LoD [7]. In comparison to our research, Leite et al. were referring to the overall building model or the combination of building elements while experimenting with the LoDs, whereas in this paper, the detailing and experiments are conducted per the individual families.

In the same context, van Berlo and Bomhof has analyzed 35 building models (where each comprises multiple building elements), taking into account different ratios between volume, triangles, space areas, and the number of properties, in an attempt to find a relationship between the different LODs [8]. However, the authors did not find any pattern for the increase of detailing across the LODs. The main reason for that is the inconsistencies and the different interpretations of the LOD specifications [9,14,40]. While some approaches use the LOD concept for describing the maturity of the overall building model, others do so only for the individual element types. van Berlo and Bomhof performed their experiments on the overall building models rather than the individual elements. By contrast, the LOD specifications provided by the AIA [34], BIMForum [6], and Trimble [10] describe the geometric and semantic information of the individual elements rather than the overall building model. On a wider scale, Wong and Ellul analyzed the geometry of 3D city models for fit-for-purpose by looking into the ratios between the number of buildings, geographic area, geometrical details, and disk size [47].

In essence, there is a significant research gap resulting in a lack of computational methods that formally specify levels of geometry on the basis of a corresponding metric, and subsequently, apply this metric on concrete building models to assess the LOG they provide.

2.6. Ensemble-learners

The process of inferring generalized patterns from a training dataset (consisting of a set of instances where their classes are known) is described as inductive inference [48]. The simplest way to analyze a training dataset is to develop a classification system that consecutively splits the data (based on feature values) in a way that groups similar classes together as much as possible. It is important to note that the metric of similarity is not pre-defined but part of the solution-finding process. Given a set of N instances where each belongs to one of K classes, a classification system can construct a set of rules through training on the set of instances. This is precisely what a decision tree performs while following a particular route, yielding a specific result [49].

A decision tree comprises a series of nodes (Boolean questions or tests), branches (results of the tests), and leaves (classification classes). Each node questions the data and splits it into two branches, eventually leading to a predicted class. In order to measure the quality of the split, two criteria are commonly used: *Information Gain*, which uses the entropy measure to split the data in a way that returns the most homogeneous branches, and the *Gini Index*, which represents the likelihood of classifying a new instance incorrectly [50]. For a given training dataset T, the Gini Index can be expressed as Eq. (1) [50]:

$$\sum\sum_{j\neq i}(f(C_i,T)/|T|)(f(C_j,T)/|T|) \tag{1}$$

Where $f(C_i,T)/|T|$ is the probability that a specific element belongs to class C_i.

Real-world data is imperfect and includes noise arising from misclassifications or inaccurate measurements. Modeling such data using one decision tree results in the generation of a long tree, which is, in this case, overfitted to the selected dataset. This is mainly because a decision tree is based on a greedy model, meaning it tries to find the most optimal decision at each step and does not consider the global optimum. Therefore, smaller trees are preferable, as they are less prone to overfitting [49], which imposes a trade-off between developing a generalized model versus its accuracy. To overcome this limitation, researchers have invented the concept of ensemble learners, which will be discussed in the next subsections.

2.6.1. Random Forest (RF)

Random forests fall into a broader category called ensemble learners [49], which generate multiple weak models and then aggregates their classifications to produce better results. As illustrated in Fig. 1, a random forest model constructs a set of decision trees and aggregates the unweighted average of their classifications (a.k.a. votes) to determine the final prediction [49].

Using methods such as bootstrap aggregating or bagging [51,52], each of the decision trees within a forest is built using a randomly selected set of features and instances. Such methods manipulate the training data to generate diverse classifiers (which makes it hard to overfit). Additionally, these methods support parallelization, making it possible to construct and train the trees within a random forest model independently from each other, which is relatively faster than other models, such as boosting, which will be discussed in more detail in the next section.

2.6.2. eXtreme Gradient Boosting (XGBoost)

From the same category as random forests, gradient boosting is an ensemble learner (combining the result of multiple weak models) [53]. The main difference between a random forest and boosting is that the former constructs decision trees independently, simultaneously, and uses an unweighted average of votes, while the latter iteratively builds and evaluates individual trees (which are usually short, a.k.a. decision stumps) and tries to learn from wrongly classified observations by adding a higher weight on them in the subsequently built trees [52,53] (the concept is illustrated in Fig. 2). An increased weight represents an increased contribution of a class or an instance to the loss function. Then, as boosting cannot be parallelized (the weights used for each tree are dependent on the results of the previously constructed tree), it takes much longer to train than a random forest.

Numerous popular boosting-based algorithms have recently been developed, including Adaptive Boosting (AdaBoost) [53] and eXtreme Gradient Boosting (XGBoost) [17]. AdaBoost is follows the weighting approach discussed previously. However, XGBoost (the currently dominant algorithm [18,54]) defines a loss function, and while iteratively constructing new trees, it focuses on minimizing that loss function. XGBoost can be expressed as Eq. (2) [17]:

$$\hat{y}_i = \phi(x_i) = \sum_{k=1}^{K} f_k(x_i), f_k \in F \tag{2}$$

Where f_k represents an independent decision tree, F is the space of trees, x_i represents the independent variables, and K are the additive functions. The goal is to minimize Eq. (3) [17]:

$$\mathcal{L}(\phi) = \sum_i l(\hat{y}_i, y_i) + \sum_k \Omega(f_k) \tag{3}$$

Where $\mathcal{L}$ is a loss function and Ω is a penalty value representing the complexity of the model by taking into account the number and score of leaves.

3. Methodology

The hypothesis of this paper is that the detailing of the individual elements at the different LOGs can be correlated with multiple geometric features. These features form the basis for formally assessing the geometric complexity of a given model. Thereby, the LOG of building elements can be identified through analyzing the detailing patterns of the extracted features across the LOGs.

As depicted in Fig. 3, the proposed approach consists of two main steps. First, a LOG dataset is modeled according to the most common LOD specifications (described in detail in Section 3.1). The dataset generation took into account modeling different kinds of building elements as well as additional cases for including openings and reinforcement. Afterwards, multiple geometry processing algorithms are performed to extract the most prominent features representing the detailing of each building element. The result is a dataset of geometric features for diverse building elements at the LOGs 200–400.

The second step describes the process of classifying the LOG of a given element that was not part of the training set (a new element). The geometric features of the new element are extracted in a similar way to the dataset generation. The individual features are then compared to the features available in the dataset to classify the LOG of the new element. This step represents the actual application of the developed approach for classifying the elements of a BIM model provided by the end-user. The complete framework is discussed in detail in the next subsections.

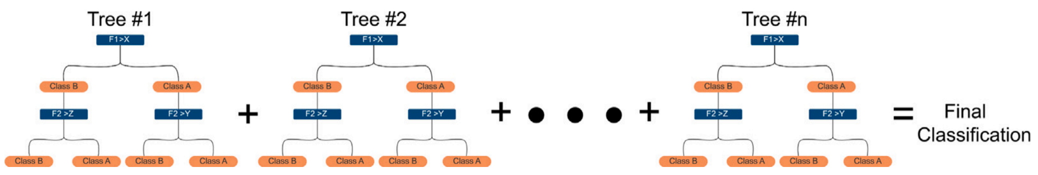


Fig. 1. Random Forest (RF) Schematic Representation: it consists of multiple decision trees, where the unweighted average of their classifications is calculated to decide on the final classification.

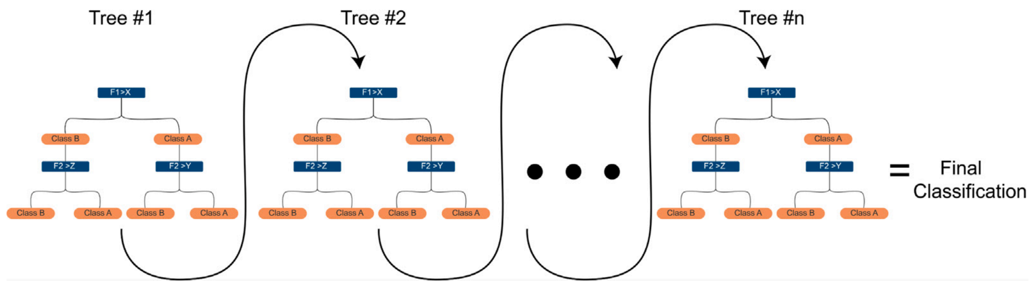


Fig. 2. eXtreme Gradient Boosting (XGBoost) Schematic Representation: it builds decision trees consecutively and tries to learn from wrongly classified observations by adding a higher weight on them in the subsequently built trees.

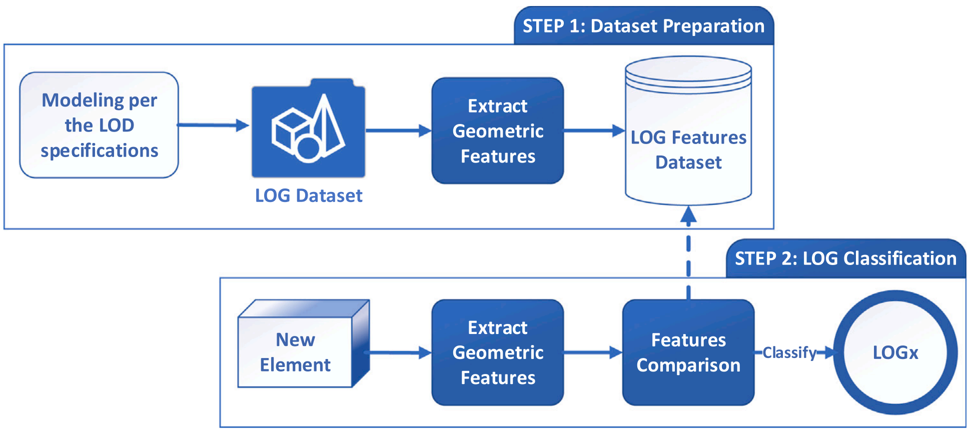

STEP 1: Dataset Preparation
Modeling per the LOD specifications
LOG Dataset
Extract Geometric Features
LOG Features Dataset
STEP 2: LOG Classification
New Element
Extract Geometric Features
Features Comparison
Classify
LOGx

Fig. 3. The developed approach presented in this paper consists of two main steps: (1) generation of a dataset containing the geometric features representing the complexity of the individual building elements across the LOGs, and (2) classification of new building elements in which their LOG is unknown.

3.1. Modeling according to the LOD specifications

In this study, the BIMForum's LOD specification [6] and Trimble's Project Progression Planning [10] were comprehensively reviewed and followed during the modeling of different families on multiple LODs. In addition to the authors' practical experience, the combination of the mentioned specifications was followed. Although the BIMForum's definitions are descriptive for many building elements, they are, in many cases, vague in describing the progression of the geometric detailing. Despite the fact that the specification is prepared in a way that visualizes the newly added parts in every LOD, the graphical illustrations for many elements are missing or inconsistent and ambiguous. For example, when modeling a staircase, information regarding the riser count and height should be available starting from LOD 300 (per the text description). However, the graphical illustration at LOD 200 already includes these information. Whereas, in Trimble's specification, for this particular case, the graphical illustration reflects the available information clearly. Therefore, it was necessary to use both specifications.

According to the LOD specifications, LOG 100 (conceptual model) is limited to a generic representation of the building, meaning no shape information or geometric representation is provided. At LOG 200 (approximate geometry), elements are represented by generic placeholders depicting the overall area reserved by their volume. At LOG 300 (precise geometry), the elements' main shape is refined, showing the fundamental detailing required for describing the element type. Next, at LOG 350 (construction documentation), any necessary parts for depicting the connections with other elements that are attached or connected are additionally modeled. Modeling these parts, like supports and connections, is crucial for the coordination with different domain experts. Finally, at LOG 400, elements and their connections are fully detailed, providing the accuracy required for fabrication, assembly, and installation. LOG 500 represents the field verified model state, but in terms of design and detailing, it is the same as LOG 400.

The modeling process followed to generate the dataset has focused on the LOGs 200–400. To have confidence in how to model the families, those which are associated with both textual description and visual illustration were modeled first. Afterwards, we expanded the dataset size by making use of the available BIM objects libraries[2] In this regard, the families were downloaded and adjusted to fit the requirements of the different LOGs. In total, the modeled dataset includes 408 objects (102 families at four LOGs). A complete list of the modeled family

[2] www.bimobject.com, http://www.nationalbimlibrary.com, http://market.bimsmith.com, http://www.revitcity.com, http://www.familit.com, http://www.arcat.com.

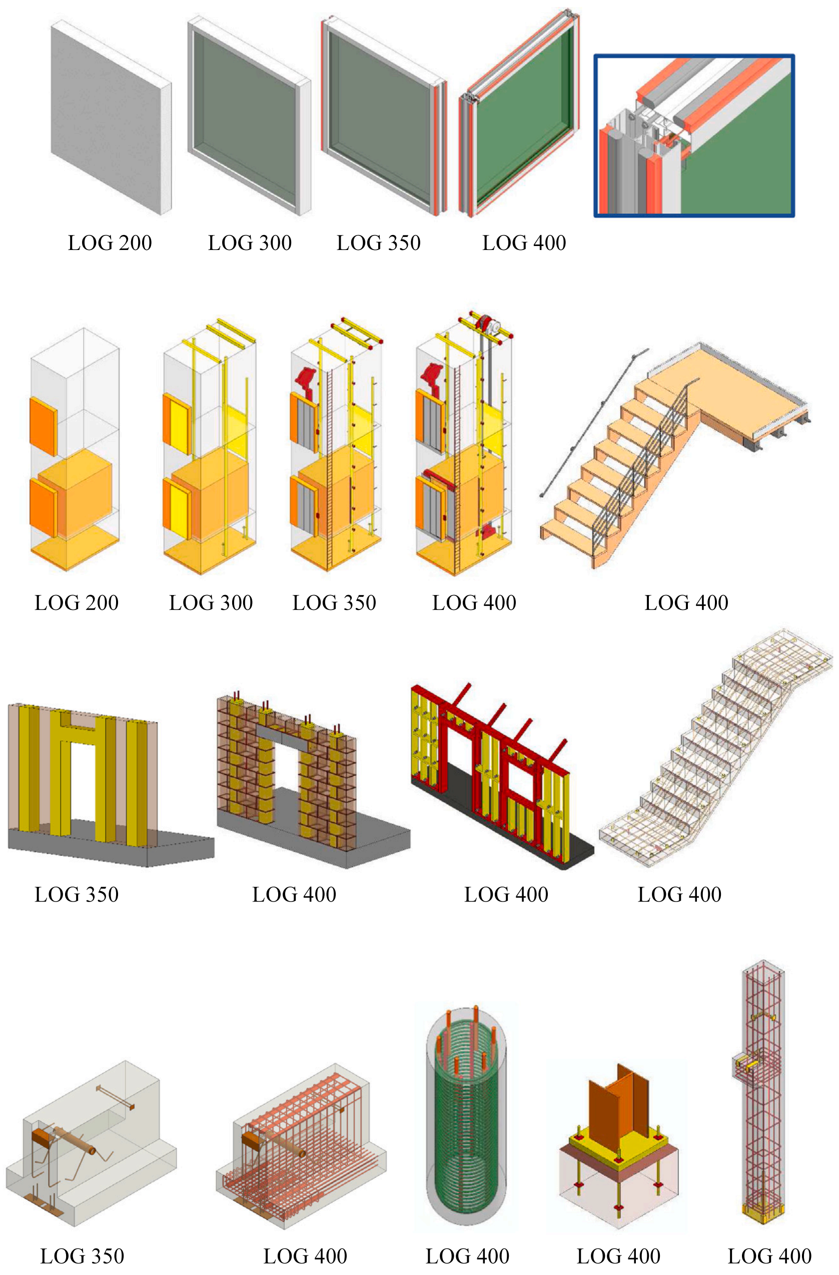


Fig. 4. LOG Dataset: A small selection of the building elements at different LOGs.

names is provided in Fig. 5. Fig. 4 shows a selected set of families on multiple LOGs from the modeled dataset. Additionally, the part of the dataset for which the authors possess full ownership is provided as open data to the public.[3]

While modeling the different families, the necessary time for modeling each LOG of each family was measured. The modeling process was conducted by two domain experts, who were responsible for the measurements. The experts are trained designers who work in an architectural office. They have a clear understanding of the LOD concept and sufficient experience in modeling families. The time starts after discussing and deciding which geometric features should be included in each family to fulfill the descriptions provided by the LOD specifications and ends after modeling all features. The aim is to investigate the necessary modeling effort associated with detailing the families from one LOG to the subsequent one.

[3] http://u.pc.cd/6fXctalK.

Fig. 6 presents the resultant time measurements from modeling the entire dataset. The figure shows the minimum, maximum, and average necessary time (in minutes) for modeling the building elements at each LOG. Modeling elements at LOG 200 required between two and 40 min. Detailing the elements further to LOG 300 utilized two to threefold of the time at LOG 200. That is mainly because modeling the outer shape at LOG 300 needs to describe the overall shape's dimensions precisely. When modeling connections with the surrounding elements at LOG 350, the necessary time increases to be between four and seven-fold the time at LOG 200. Finally, as LOG 400 demands fabrication-level detailing, the necessary time doubles, reaching eight to 15-fold compared to LOG 200. When comparing the increase in the time between subsequent LOGs, we observe that it ranges between 1.88–2.80-fold.

3.2. Analysis and extraction of LOG features

Typically, shapes having more numerous or smaller features can be viewed as more detailed. The challenge in identifying the LOG through

1. Ramp	2. Portable Water Storage Tank	3. Wood Stair (variant 1)
4. Escalator (variant 1)	5. Domestic Water Equipment	6. Fireplace (variant 1)
7. Escalator (variant 2)	8. Domestic Water piping	9. Fireplace (variant 2)
10. Rectangular Pier (Reinforced Concrete)	11. Plumbing Fixtures	12. Sink
13. Cylindrical Pier (Reinforced Concrete)	14. Stormwater Drainage Piping	15. Balcony Railing
16. Helical Pile	17. Stormwater Drains	18. Heating Pipe Fittings
19. Exterior Wall (Brick)	20. Fuel Storage Tanks	21. Concrete Batch Plant
22. Exterior Wall (Wood)	23. Heat Generation	24. Steel Base Plate
25. Masonry Framing (variant 1)	26. Supply Air	27. Fire mains
28. Masonry Framing (variant 2)	29. Water-Based Fire Suppression	30. Bathtub
31. Cold-Form Metal Framing	32. Packaged Generator Assembly	33. Chimney (Brick)
34. Precast Structural Inverted T Beam (Concrete)	35. Electrical Service Entrance	36. Tube Light
37. Roof (Clay)	38. Power Distribution	39. Cables System
40. Roof (Wood Shingles)	41. Lighting Fixtures	42. Toilet
43. Floor Structural Frame	44. Metal Building Systems - Primary Framing	45. Office Desk (variant 1)
46. Multilayers Slab (Reinforced Concrete)	47. Metal Building Systems - Secondary Framing	48. Office Desk (variant 2)
49. Precast Structural Inverted T Beam (Concrete)	50. Electric Distribution System	51. Laundry Sink
52. Precast Structural Column (Concrete)	53. Concrete Column Formwork	54. Office Chair (variant 1)
55. Steel Framing Column	56. Concrete Slab Formwork	57. Office Chair (variant 2)
58. Steel Framing Beam	59. Highway Bridges Precast Structural Girder (Concrete)	60. Double Electrical Door
61. Steel Framing Bracing Rods	62. Slide Window	63. Bed Side Drawer
64. Steel Joists	65. Architectural Column	66. Water Boiler
67. Wood Floor Trusses	68. Open Balcony	69. Roof Ladder
70. Precast Structural Double Tee (Concrete)	71. Covered Balcony	72. Sofa
73. Precast Structural Stairs (Concrete, variant 1)	74. Curtain Wall	75. Sliding door
76. Precast Structural Stairs (Concrete, variant 2)	77. Garage Door	78. HVAC System
79. Metal Walkways	80. Lamp (variant 1)	81. Trefoil Round Arch Window
82. Precast Wall Construction (Concrete)	83. Lamp (variant 2)	84. Trefoil Round Arch Door
85. Exterior Window (variant 1)	86. Window Shading	87. Ventilation System
88. Exterior Window (variant 2)	89. Sliding door	90. Wardrobe
91. Interior Door (variant 1)	92. Rotate Door (variant 1)	93. Inline Pump
94. Exterior Door (variant 1)	95. Rotate Door (variant 2)	96. Roof Hatch
97. Elevator (variant 1)	98. Spiral Metal Stair	99. Tube System
100.Elevator (variant 2)	101.Reinforced Wall	102. Wood Stair (variant 2)

Fig. 5. Families Dataset: list of the modeled family names.

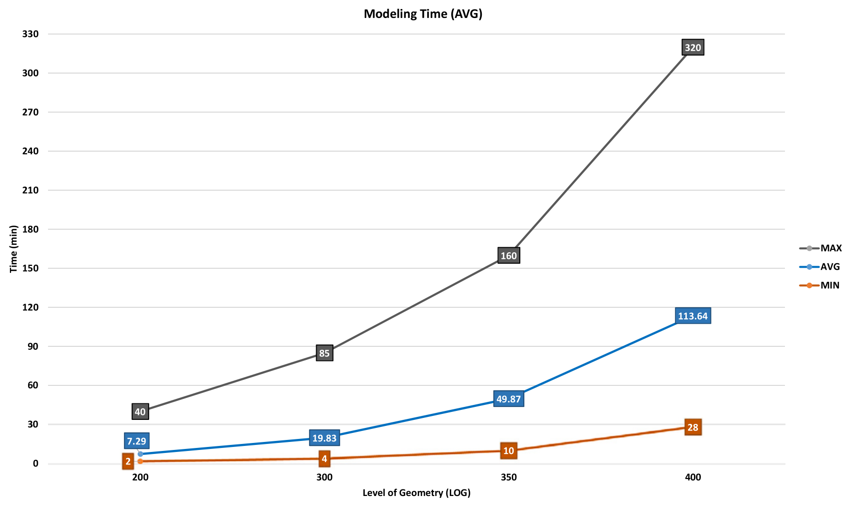


Fig. 6. LOG Dataset: an investigation of the necessary modeling time when detailing building elements from LOG 200–400. The figure shows the minimum, maximum, and average time (in minutes).

analyzing the geometric features lies in deducing a standard pattern (a metric) that describes the individual LOGs. The simplest geometric metrics can be based on the total number of vertices, faces, and edges. However, an increased number of these features does not necessarily mean an increased detailing or higher LOG. For example, a window at LOG 200 (rectangular shape) consists of 30 vertices, 16 faces, and 78 edges, while a cylindrical column or heating tank at LOG 200 could be formed by 2,358 vertices, 4,268 faces, and 13,244 edges.

Thus, the sole consideration of vertices, faces, and edges does not provide a suitable metric. To measure the geometric detailing (i.e., LOG) of elements, the set of selected features needs to be capable of representing the geometric detailing of elements taking into account the overall shape complexity. Hence, in this paper, we propose combining the extracted results of multiple geometric features to observe various aspects of the shape's detailing. In total, we investigated the effect of detailing across the LOGs through three main aspects, which are discussed in detail in the next subsections.

3.2.1. *Basic features: Vertices, faces, and edges*

Vertices, faces, and edges represent the most fundamental ingredients for describing the detailing of any shape. In this regard, the ratio of vertices to faces is capable of providing an insight into the overall shape form. Based on our experiments, a shape with only rectangular

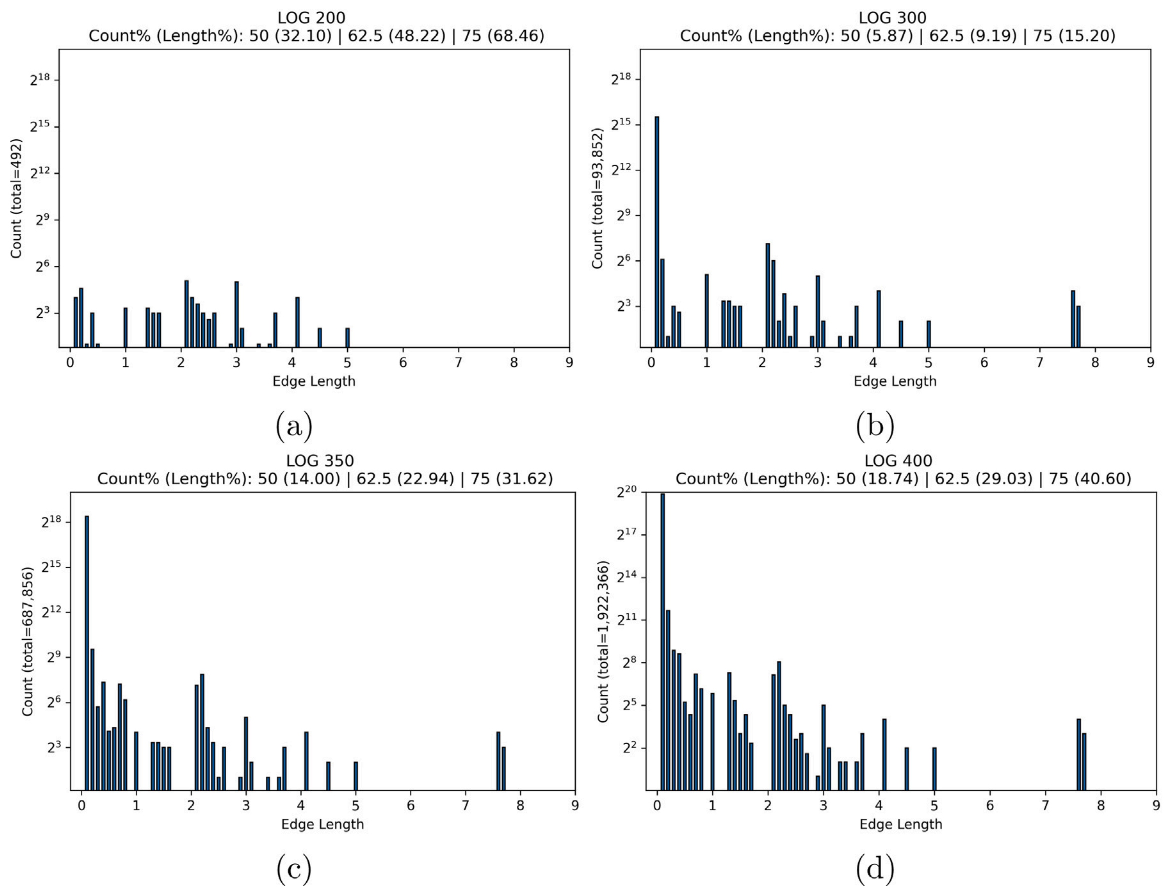


Fig. 7. Basic Geometric Features: count and percentage of edges' lengths across the LOGs for the elevator element depicted in Fig. 4. Here, the mean of the edges' length as well as the total length of 50%, 62.5%, and 75% of the edges after ordering them in an ascending order were measured. Prioritizing the short edges provides an indicator for the increase in detailing.

parts always has a ratio of two. When adding more complex parts, like screws or reinforcement, the ratio is substantially reduced.

The count and length of edges can also provide a strong description of the shape resolution and complexity [55]. When a shape comprises a low number of edges with a similar length, then the overall shape is basic and does not include high details. On the other hand, when the majority of edges are relatively short, then the shape comprises numerous complex parts. In this regard, we measure the mean of the edges' length as well as the total length of 50%, 62.5%, and 75% of the edges, after ordering them in ascending order to prioritize the short edges, as they are an indicator for the increase in detailing. Then, the measured lengths' ratio to the total edges' length is calculated at the different LOGs. For example, for a particular element, the length of 75% of the edges could represent 60% of the length of the total edges at LOG 200, while the length of 75% of the edges represents 10% at LOG 400.

Fig. 7 presents the different lengths of edges on the x-axis and the total count of each length on the y-axis for the elevator element depicted in Fig. 4. The edges' lengths (on the x-axis) were rounded to the first decimal place and grouped. The edges' counts (on the y-axis) are shown on a logarithmic scale (to the base of two) to highlight the different lengths. At LOG 200, the total count of edges is 492. Here, we can notice that the edges' counts are relatively comparable across the lengths of zero to five and mostly dense in the middle. At LOG 300, the total count of edges became 93,852 (19-fold the count at LOG 200). Although numerous relatively long edges were added, the majority of the edges are short. The increase in the count and length of edges across the LOGs 350 and 400 follows a similar pattern.

Additionally, Fig. 7 lists the statistical percentages of the edges' counts and lengths. Such statistics highlight the overall geometrical detailing. In more detail, at LOG 200, the shape is expected to be an approximation, represented by bounding boxes. Therefore, the length of 62.5% and the length of 75% of the edges equals 48.22% and 68.46% of the overall length, respectively, which are relatively high. Whereas, at LOG 300, the shape is refined further to represent a precise shape, resulting in numerous additional short edges. Accordingly, the length of the edges is much shorter than at LOG 200. Next, at LOGs 350 and 400, connections and additional geometric details (for example, for fabrication or even vendor-specific details) are modeled, gradually increasing the length of the statistical percentages.

3.2.2. *Sharp edges and feature lines*

Feature lines identify the most prominent surface characteristics of a geometric shape [56]. The extraction of these lines has been intensively researched in various domains, including the analysis of medical data [57] and point clouds [58]. The fundamental description of feature lines is the local extrema of principal curvatures along with corresponding principal directions [56]. In other words, the angle between the two normal vectors of adjacent triangles is measured, and when the angle is sharp (the surface curvature is changing), then the edge is considered as a feature edge. Finally, the detected edges form the shape's feature lines.

We extract and count the sharp edges as well as the number of surface patches bound by these edges. Fig. 8 shows a stair and a window at LOG 200. Here the sharp edges are marked with a red color.

3.2.3. *Diameter-based segmentation*

In this approach, the shape is segmented into smaller meaningful pieces based on the change in its diameter [25]. The segmentation is based on measuring the Shape-Diameter Function (SDF) at every

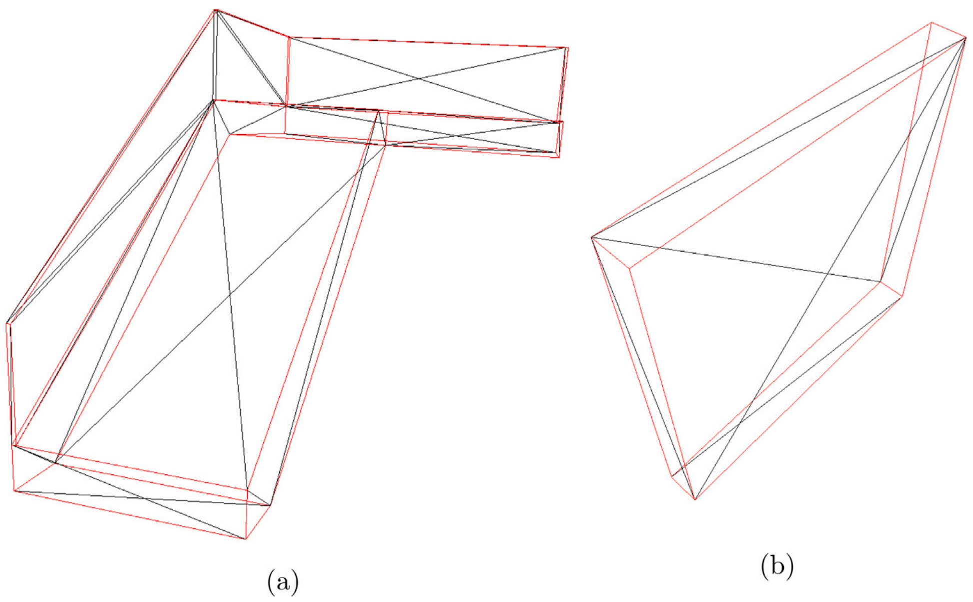


Fig. 8. Sharp Edges: an example of a stair and window at LOG 200. The edges marked with red represent the extracted sharp edges.

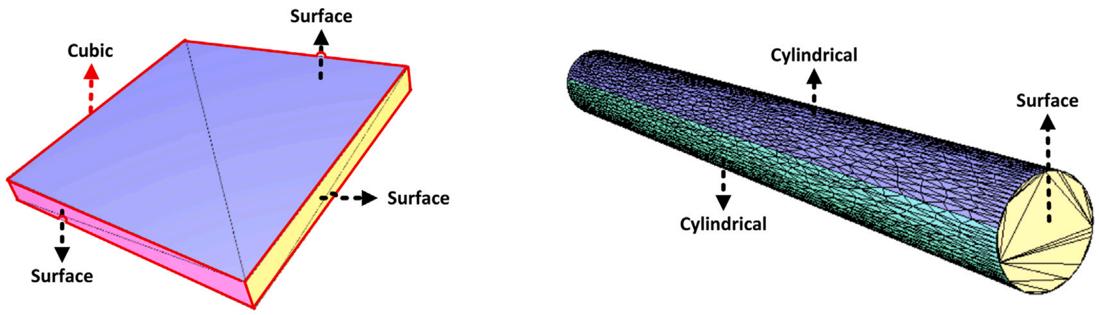


Fig. 9. Diameter-based Segmentation: two examples highlighting the results of segmenting the building elements. The colors here represent the individual visible segments.

point, where the change of an SDF value from a point to its neighbors determines whether there is a new segment. Let M be a triangulated mesh surface of any building element. The SDF is defined as the scalar function on the surface $fv : M \rightarrow \mathbb{R}$, representing the diameter at every neighbor point $p \in M$. The SDF provides an effective link between the object's volume to its surface. The algorithm provided by Shapira et al. [25] applies clustering on the facets according to their corresponding SDF values. Afterwards, the dihedral-angle and concavity of the surfaces is taken into account to produce the final segments.

This kind of segmentation provides additional insights into the complexity of the parts that are forming the building element. Therefore, we count the segments, measure their area, and evaluate their shape (flat surfaces, cubic, or cylindrical). Segments with similar shapes are grouped, and the ratio of their count and area is used to characterize the form and complexity of the overall shape. For example, a window at LOG 200 comprises few surfaces and cubic segments. Whereas, in the case of a tube system at LOG 200, it comprises few surfaces and cylindrical segments. Additionally, at LOG 400, numerous smaller segments with diverse shapes are typically added. Fig. 9 shows two examples, highlighting the individual segments.

To highlight the benefit of counting and grouping the shape of the extracted segments, Fig. 10 shows the segments' shapes on the x-axis and the total count of segments on the y-axis for the elevator element depicted in Fig. 4. Besides increasing the total number of segments, these statistical calculations provide additional insights into what kind of detailing was added at each LOG. Additionally, this information facilitates identifying the shape characteristics. For example, based on our evaluations, when the count of the cylindrical segments is low and represents more than ~50% of the overall area, the overall shape has a high probability of having a cylindrical overall shape (a pipe, for example). Moreover, rectangular and complex shapes (such as a window and a stair) at the LOGs 350 and 400 are composed of a high number of cylindrical segments representing less than ~40% of the overall area, which indicates the presence of screws and additional detailing parts. When reinforcement is modeled, then the number of cylindrical segments is relatively high (~50%–80%), while their aggregated area is less than ~40% of the overall area.

3.2.4. *Features dataset*

The discussed geometric features above were extracted for the LOG dataset presented in Section 3.1. Additionally, multiple ratios were calculated to capture any positive or negative correlations among the features, including the average area per surface patch and per segment, as well as the average number of vertices per face, patch, and segment. Finally, the extracted features were normalized to make the features correspond to the elements' geometric complexity regardless of their total area or total length of edges.

To get an overview of the degree of association between the extracted features, a pair-wise Pearson correlation coefficient (PCC) [59] was calculated. PCC measures the level of linear correlation between two variables. Accordingly, these coefficients are leveraged during the ensemble models' training to filter and optimize which features are selected for the training process. The features that prove a linear correlation (0.8 – 1 PCC) are considered the first candidates to be dropped

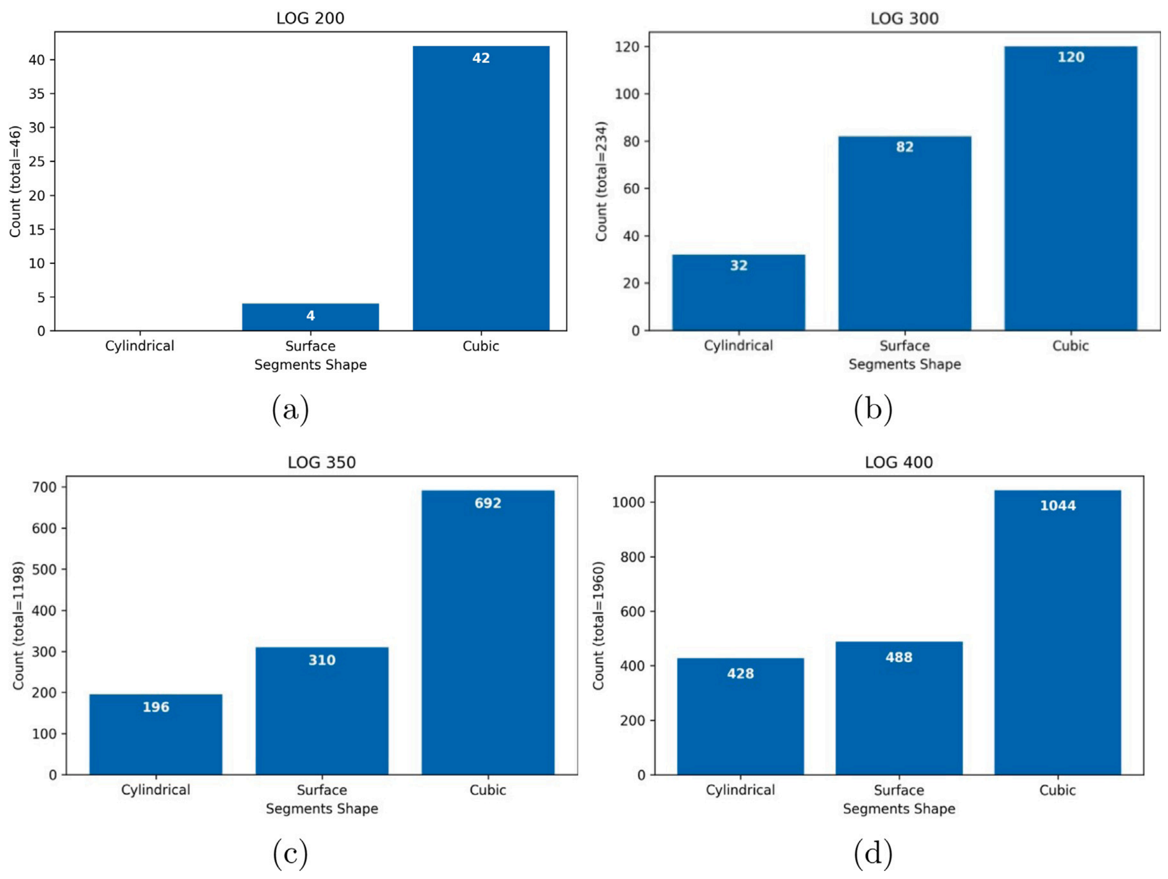


Fig. 10. Diameter-based Segmentation: statistical analysis of the extracted segments based on their shape (surface, cubic, and cylindrical). These calculations were extracted from the elevator element depicted in Fig. 4. The x-axis lists the segments' shape and the y-axis shows the total count of segments.

Table 1
Features Dataset: an example of the selected features for training the ensemble models. The examples shown here belong to a *Brick Wall* and a *Reinforced Concrete Pier* across the LOGs.

LOG	$\frac{\text{Vertices}}{\text{Faces}}$	$\frac{\text{Vertices}}{\text{Patches}}$	$\frac{\text{Vertices}}{\text{Segments}}$	$\frac{\text{Faces}}{\text{Patches}}$	$\frac{\text{Area}}{\text{Patches}}$	$\frac{\text{Area}}{\text{Segments}}$	$\frac{(\%)}{\text{SharpEdges}}$	$\frac{\text{SharpEdges}}{\text{Area}}$	$\frac{\text{SharpEdges}}{\text{Vertices}}$
200	1.8	4.5	4.5	2.5	12.5	12.5	37.5	2.78	1
300	1.8	4.5	4.5	2.5	6.25	6.25	37.5	1.39	1
350	0.73	62.53	62.53	85.96	0.88	0.88	12.58	0.02	0.59
400	0.64	72.1	74.32	112.57	0.031	0.03	8.81	0.001	0.45
200	0.67	1.34	8	2	16.67	100	33.34	8.34	1.5
300	0.57	1.6	16	2.8	10	100	28.57	4.17	1.5
350	0.53	1.65	37.45	3.11	0.2	4.55	27.44	0.078	1.56
400	0.50	1.43	172.17	2.84	0.02	2.08	28.40	0.007	1.69

LOG	(%) Mean edges length	(%) 75% edges length	(%) Mean segments area	(%) 62.5% segments area	Cylindrical segments area	(%) Cylindrical segments count	(%) Cylindrical segments area
200	2.08	76.97	12.85	13.81	0	0	0
300	1.04	59.72	6.34	23.30	0	0	0
350	0.006	33.89	0.89	12.73	2.39	22.81	4.24
400	0.0001	35.92	0.03	12.024	25.74	19.41	25.65
200	5.56	55.23	100	100	3.78	100	100
300	2.38	38.48	100	100	3.93	100	100
350	0.04	18.77	4.56	3.74	4.41	100	100
400	0.004	3.25	2.09	12.46	5.75	100	100

to simplify the vector representation of each element. Additionally, the PCCs were combined with the features' importance (shown in Fig. 12) to decide which features could be dropped. Such filtering is important when training tree-based models since unimportant features could construct weak trees that could then affect the model's accuracy. The features dataset included 22 features per element before filtering and 16 features after filtering. Table 1 depicts sample features of two building elements, a *Brick Wall* and a cylindrical *Reinforced Concrete Pier*, across the LOGs.

4. Classification of LOG

The analysis of the extracted geometric features indicated multiple patterns that are helpful in identifying the LOG. Identifying which class (i.e., LOG) an observation (i.e., the features representation of an element) belongs to is a classification problem. The manual classification of the LOG from the extracted features is an unfeasible task due to the large number of features and the heterogeneity of the different families. Hence, in this paper, we propose classifying the LOG of building elements using RF and XGboost, tree-based ensemble-learning models. Such models are popular nonlinear predictive models. In the following, we compare the approaches regarding their performance for the problem at hand.

4.1. Models training setup

The features dataset presented in Section 3.2.4 was split into training and testing sets with a ratio of 80% (326 elements) and 20% (82 elements), respectively. Splitting the dataset involved taking into account the different classes (LOGs 200, 300, 350, and 400) and the type of families (to ensure a sufficient diversity). Some families at a specific LOG were only available in one of the sets.

Training an ensemble model involves tuning its hyperparameters to make its architecture more suitable for the used features. Such activity highly influences the model's accuracy and capability to generalize from individual observations. During the tuning of parameters, the best performing values are identified by searching through a range of values and evaluating all the possible combinations. In order to evaluate the performance of each set of parameters, we use a technique called k-fold cross-validation (K-foldCV) [60]. K-foldCV iteratively splits the training set into k smaller sets. For each k of the folds, the model is trained using $k-1$ while tested on the remaining part to evaluate the model's accuracy during training. Afterwards, the performance of the final model is measured by validating it against the test set. A too-large k-fold means that a low number of samples is validated in every iteration. Therefore, based on multiple experiments and given the diversity and size of the dataset used in this study, 5-fold cross-validations were performed to cover enough samples in every iteration.

The interpretation of the classifications resulting from the ensemble models is critical to understand the contribution of the individual features. Therefore, we use the SHapley Additive exPlanations (SHAP) [61] approach to assign each feature an importance value for each LOG class, providing detailed features' importance. SHAP is based on game theory [62] and local explanations [63]. Assume an ensemble model that is trained on all feature subsets $S \subseteq F$, where F is the set of all features. The contribution of each feature ϕ_i on the model output is computed based on its marginal contribution compared to the rest of the features. The computation of SHAP values for a withheld feature can be represented as Eq. (4) [61]:

$$\phi_i = \sum_{S \subseteq F \setminus \{i\}} \frac{|S|!(|F| - |S| - 1)!}{|F|!} \left[f_{S \cup \{i\}}(x_{S \cup \{i\}}) - f_S(x_S) \right] \tag{4}$$

Where x_S represents the values of the input features in the set S. Additionally, $f_{S\cup\{i\}}(x_{S\cup\{i\}})$ represents the model's classification when trained on all features without the withheld feature, and $f_S(x_S)$ is the model's classification when trained on the withheld feature (See [61] for more details).

4.2. RF model training

The hyperparameters that require tuning when training a RF model are shown in Table 2. The figure includes the range of values examined to find the best combination of parameters, which are also shown in a separate column (with *Selected* as a title). The selection of parameters was based on evaluating the model's accuracy while examining all the possible combinations. In addition to these parameters, the RF model is configured to bootstrap the training data while constructing the decision trees. Bootstrapping brings more variation to the training samples through shuffling and random filtering. Finally, the *Gini Index* is selected as the function to measure the quality of each split.

To explore the structure of the decision trees comprising the RF model, Fig. 11 shows a randomly selected tree out of the 170 trees in the forest. Each tree in the forest is built differently, where different order and feature types are used to split the dataset at every node. For each tree, the upper nodes split the data into smaller clusters, while the leaves classify the data into different LOG classes, specifying the confidence of every classification. The confidence of final classifications is represented using the *Gini Index* measure, where zero means 100% confidence and one means 100% uncertain. All nodes are colored according to their respective LOG class, and the color saturation reflects the classification confidence. For example, LOG 300 and 100% confidence (0.0 gini) is colored with dark green, while LOG 200 with 50% confidence (0.5 gini) is colored with light orange.

For the particular example shown in Fig. 11, 211 elements out of the 326 were selected using boosting (other trees are constructed using a differently selected set of elements). In this example, the percentage of the cylindrical segments to all segments is the first feature splitting the training set to LOG 200 (75 elements) and LOG 350 (136 elements). Then, the total area ratio to all segments splits the data at the upper branches to LOG 300 (42 elements) and 200 (33 elements), whereas, the percentage of the cylindrical segments splits the elements at the lower branch to LOG 350 and 300. This process of splitting the dataset continues until reaching the leaves. At the leaves, a final classification is predicted for each group of elements. For instance, the first blue leaf at the top is 100% confident of classifying three of the samples as LOG 350, and the third green leaf under it is 81% confident of classifying ten samples as LOG 300. To provide additional insight on the RF model structure, multiple decision trees are provided online.[4]

As shown in Fig. 11, tree-based models base their classifications on the combination of different features. Typically, the features that contribute more to determining the final classification have higher importance. Such features are present multiple times within the constructed trees and split the data with high confidence. Fig. 12 presents the importance of the features within the RF model (based on all 170 decision trees). A higher SHAP mean value implies higher importance of the corresponding features. Additionally, the bar of each feature is divided into four parts to quantify its influence on classifying each LOG. In this particular case, the top four important features involve the count and area of the extracted segments as well as the count and length of the detected sharp edges, including the resultant surface patches. On the other hand, the basic geometric features, like the count of vertices, faces, and edges, have considerably lower importance in contributing to the final classifications. Although the overall importance of the top five features is relatively higher than others, some of the other features are essential for differentiating a particular LOG from others, such as the cylindrical segments that are more present in LOG 400 than LOG 200, unless the overall shape is cylindrical.

4.3. XGBoost model training

For comparison, XGBoost was trained on the same dataset as it is one of the best performing algorithms for solving classification problems [18,54]. The hyperparameters tuned during training are shown in Table 3, including 150 decision trees, a maximum depth of four, and multiple other parameters that influence the learning process. Similarly to the RF model, choosing the model parameters was based on evaluating a range of values.

Since the concept behind XGBoost is different from RF, the structure of the decision trees is also different. The trees are built subsequently and dependently rather than simultaneously and independently. Accordingly, the leaves of every branch within each tree produce a margin value (between −1–1), contributing to the overall classification probability of each class. This process is repeated for each class to

[4] https://bit.ly/3jPJBeu.

Table 2
RF model hyperparameters, including the search ranges and the selected values.

Parameter name	Description	Search range	Selected
n_estimators	Number of trees in the forest	20–1000	170
max_depth	Maximum depth of the trees	2–20	5
max_features	Maximum number of features to consider when splitting the data at a particular node	2–8	2
min_samples_leaf	Minimum number of samples required to be at the leaf	2–6	2
min_samples_split	Minimum number of samples required to split a node	2–6	2

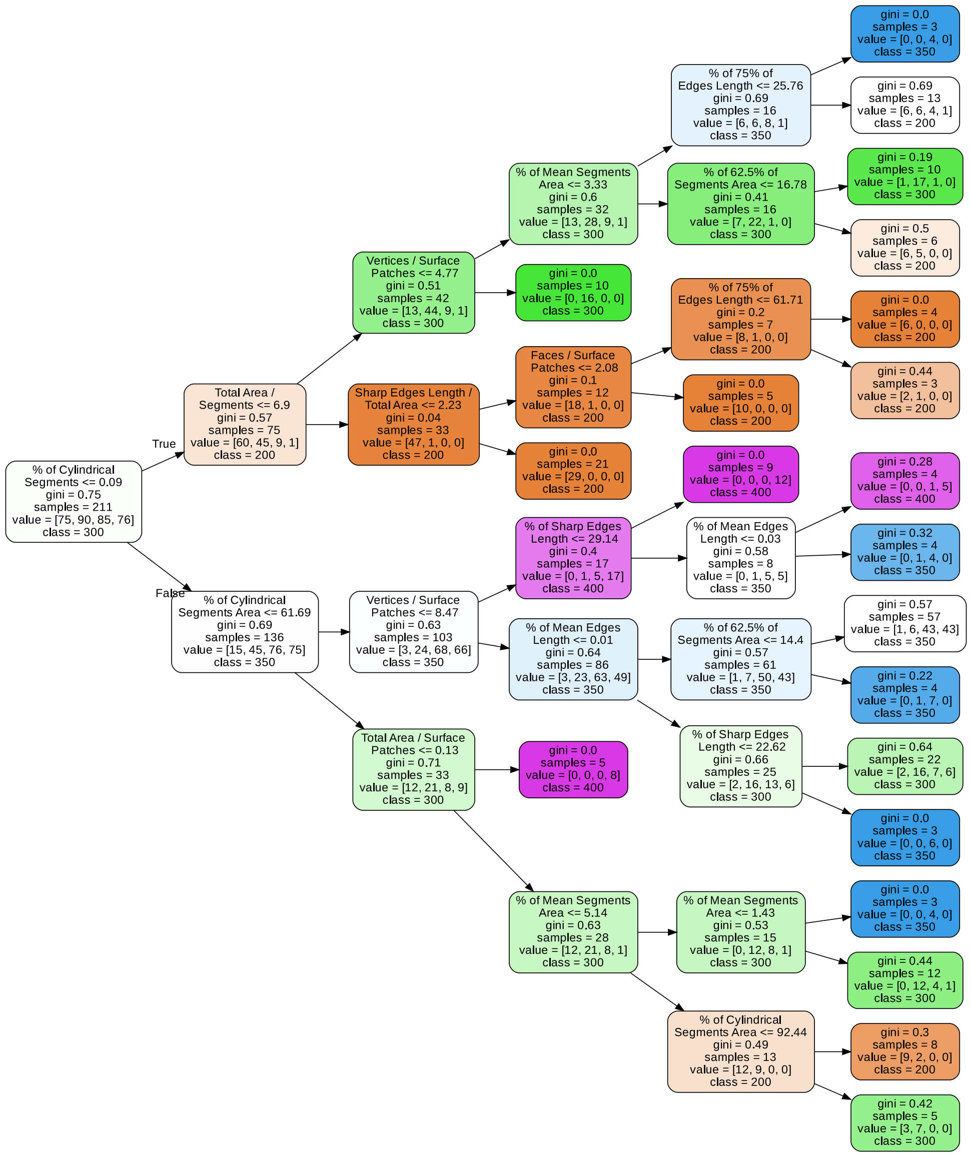

% of Cylindrical Segments <= 0.09 gini = 0.75 samples = 211 value = [75, 90, 85, 76] class = 300
True
False
Total Area / Segments <= 6.9 gini = 0.57 samples = 75 value = [60, 45, 9, 1] class = 200
Vertices / Surface Patches <= 4.77 gini = 0.51 samples = 42 value = [13, 44, 9, 1] class = 300
% of Mean Segments Area <= 3.33 gini = 0.6 samples = 32 value = [13, 28, 9, 1] class = 300
% of 75% of Edges Length <= 25.76 gini = 0.69 samples = 16 value = [6, 6, 8, 1] class = 350
gini = 0.0 samples = 3 value = [0, 0, 4, 0] class = 350
gini = 0.69 samples = 13 value = [6, 6, 4, 1] class = 200
% of 62.5% of Segments Area <= 16.78 gini = 0.41 samples = 16 value = [7, 22, 1, 0] class = 300
gini = 0.19 samples = 10 value = [1, 17, 1, 0] class = 300
gini = 0.5 samples = 6 value = [6, 5, 0, 0] class = 200
gini = 0.0 samples = 10 value = [0, 16, 0, 0] class = 300
Sharp Edges Length / Total Area <= 2.23 gini = 0.04 samples = 33 value = [47, 1, 0, 0] class = 200
Faces / Surface Patches <= 2.08 gini = 0.1 samples = 12 value = [18, 1, 0, 0] class = 200
% of 75% of Edges Length <= 61.71 gini = 0.2 samples = 7 value = [8, 1, 0, 0] class = 200
gini = 0.0 samples = 4 value = [6, 0, 0, 0] class = 200
gini = 0.44 samples = 3 value = [2, 1, 0, 0] class = 200
gini = 0.0 samples = 5 value = [10, 0, 0, 0] class = 200
gini = 0.0 samples = 21 value = [29, 0, 0, 0] class = 200
% of Cylindrical Segments Area <= 61.69 gini = 0.69 samples = 136 value = [15, 45, 76, 75] class = 350
Vertices / Surface Patches <= 8.47 gini = 0.63 samples = 103 value = [3, 24, 68, 66] class = 350
% of Sharp Edges Length <= 29.14 gini = 0.4 samples = 17 value = [0, 1, 5, 17] class = 400
gini = 0.0 samples = 9 value = [0, 0, 0, 12] class = 400
% of Mean Edges Length <= 0.03 gini = 0.58 samples = 8 value = [0, 1, 5, 5] class = 350
gini = 0.28 samples = 4 value = [0, 0, 1, 5] class = 400
gini = 0.32 samples = 4 value = [0, 1, 4, 0] class = 350
% of Mean Edges Length <= 0.01 gini = 0.64 samples = 86 value = [3, 23, 63, 49] class = 350
% of 62.5% of Segments Area <= 14.4 gini = 0.57 samples = 61 value = [1, 7, 50, 43] class = 350
gini = 0.57 samples = 57 value = [1, 6, 43, 43] class = 350
gini = 0.22 samples = 4 value = [0, 1, 7, 0] class = 350
% of Sharp Edges Length <= 22.62 gini = 0.66 samples = 25 value = [2, 16, 13, 6] class = 300
gini = 0.64 samples = 22 value = [2, 16, 7, 6] class = 300
gini = 0.0 samples = 3 value = [0, 0, 6, 0] class = 350
Total Area / Surface Patches <= 0.13 gini = 0.71 samples = 33 value = [12, 21, 8, 9] class = 300
gini = 0.0 samples = 5 value = [0, 0, 0, 8] class = 400
% of Mean Segments Area <= 5.14 gini = 0.63 samples = 28 value = [12, 21, 8, 1] class = 300
% of Mean Segments Area <= 1.43 gini = 0.53 samples = 15 value = [0, 12, 8, 1] class = 300
gini = 0.0 samples = 3 value = [0, 0, 4, 0] class = 350
gini = 0.44 samples = 12 value = [0, 12, 4, 1] class = 300
% of Cylindrical Segments Area <= 92.44 gini = 0.49 samples = 13 value = [12, 9, 0, 0] class = 200
gini = 0.3 samples = 8 value = [9, 2, 0, 0] class = 200
gini = 0.42 samples = 5 value = [3, 7, 0, 0] class = 300

Fig. 11. RF model: showing one decision tree out of 170. The nodes are colored according to their respective LOG class, and the color saturation reflects the classification confidence. For example, LOG 300 with 100% confidence is colored with dark green, while LOG 200 with 50% confidence is colored with light orange.

represent the probability that a path through each tree classifies each class with a particular value. In the end, the sum of values from the subsequent trees provides the overall classes probabilities (see Fig. 13).

Fig. 14 presents more insights on the overall features' importance for the XGBoost model. In this regard, the top four features are similar to the RF model, involving the count and area of the extracted segments as well as the count and length of the detected sharp edges, including the resultant surface patches. However, those features have different SHAP mean values as well as different proportions for the LOG classes. Additionally, the rest of the features are ordered differently from the RF model. In general, we observe that the XGBoost model relies on fewer features than the RF model to make the final classification.

4.4. Evaluation of RF and xgboost models

The performance of the developed ensemble models was evaluated on a new set of elements (a test dataset that consists of 82 elements, entirely disjoint from the training set). The performance metrics are described as precision, recall, and F1-Score. Precision describes the model performance in positive predictions while considering false positives.

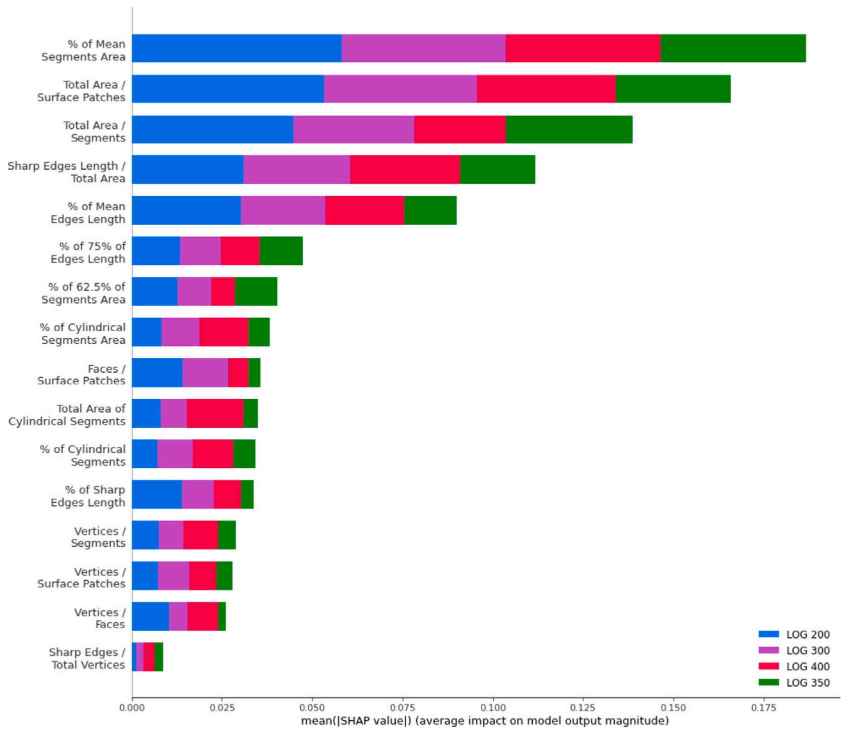


Fig. 12. Features' importance for the RF model (all trees): using SHAP mean values as an indicator, a higher value implies higher importance of the corresponding feature.

Table 3
XGBoost model hyperparameters, including the search ranges and the selected values.

Parameter name	Description	Search range	Selected
n_estimators	Number of trees in the forest	20–1000	150
max_depth	Maximum depth of the trees	2–20	4
learning_rate	Step size shrinkage used in update to prevent overfitting	0.001–1	0.001
gamma	Minimum loss reduction required to make a further partition on a leaf node of the tree	0.1–1	0.54
min_child_weight	Minimum sum of instance weight (hessian) needed in a child. If the tree partition step results in a leaf node with the sum of instance weight less than min_child_weight, then the building process will give up further partitioning	0.1–1	0.8
subsample	Subsample ratio of the training instances	0.1–1	0.6

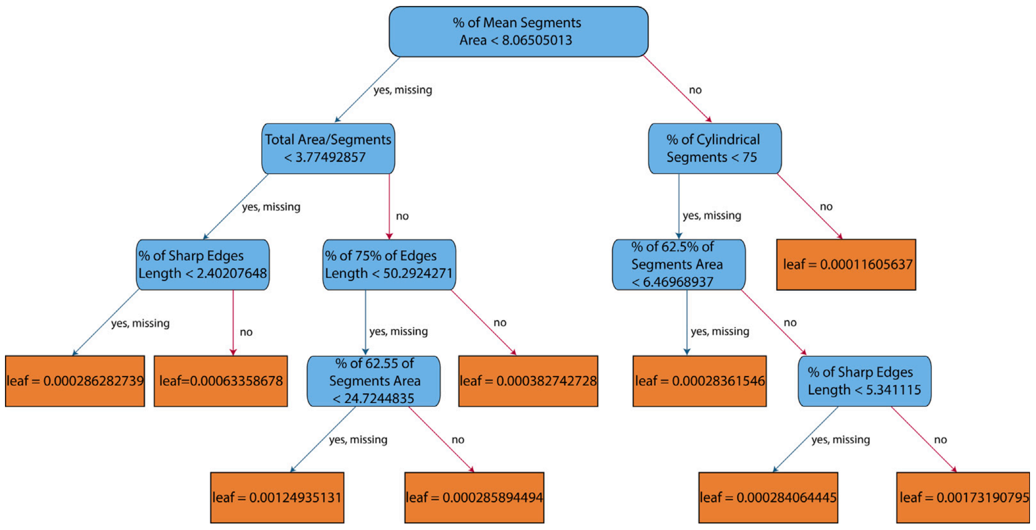


Fig. 13. XGBoost model: showing one decision tree out of 150. The leaves of every branch within each tree produce a margin value (either a positive or negative number), which contributes to the overall classification of each element when combined with the previous and subsequent trees' results.

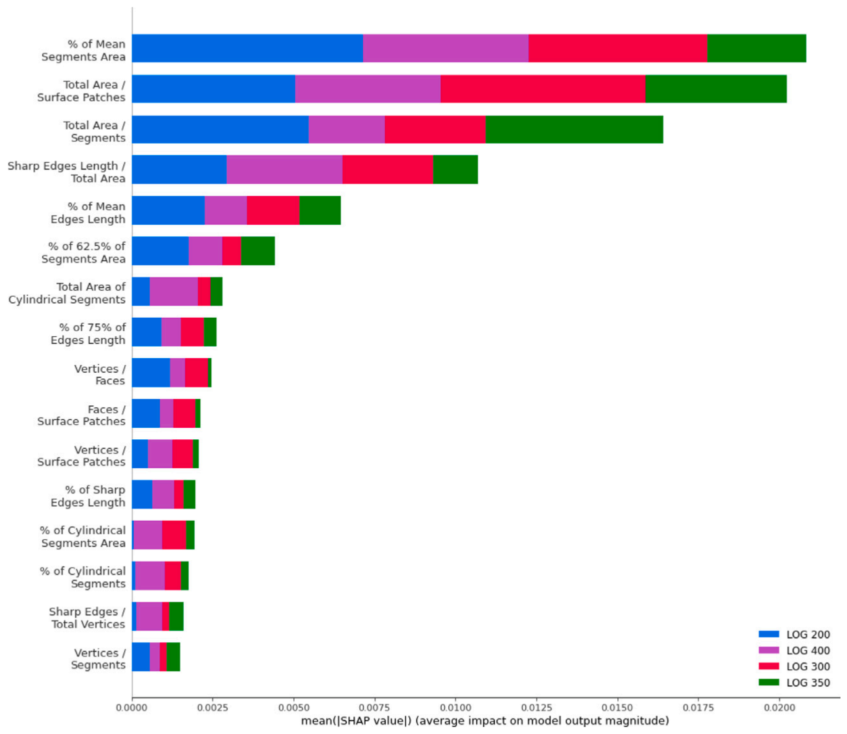


Fig. 14. Features importance for the XGBoost model (all trees): using SHAP mean values as an indicator, a higher value implies higher importance of the corresponding feature.

Table 4
Evaluation results: performance results of the RF and XGBoost models on test data. The performance metrics are described as precision, recall, and F1-Score. Precision describes the model performance in positive predictions while considering false positives. Recall incorporates false negatives instead of false positives, and F1-score provides a balance between precision and recall.

LOG	Precision		Recall		F1-score	
	RF	XGBoost	RF	XGBoost	RF	XGBoost
200	0.96	1.00	0.81	0.81	0.88	0.90
300	0.74	0.75	0.85	0.90	0.79	0.82
350	0.71	0.73	0.71	0.79	0.71	0.76
400	0.86	0.90	0.90	0.90	0.88	0.90
Accuracy					**0.83**	**0.85**
Macro avg.	0.82	0.85	0.82	0.85	0.82	0.84
Weighted avg.	0.84	0.87	0.83	0.85	0.83	0.86

Recall incorporates false negatives instead of false positives, and F1-score provides a balance between precision and recall. Table 4 presents the evaluation results of both models. Generally, the performance of both models in classifying the four LOGs is relatively close. The XGBoost outperforms RF in the precision, recall, and F1-score for all the LOGs. However, the F1-score's difference is not substantial for most LOGs (2–3% for all except LOG 350, 5%).

To understand the evaluation results in more detail, Fig. 15 shows the confusion matrix for both models depicting the difference between the actual and predicted LOGs. 68 and 70 out of 82 elements were classified correctly using RF and XGBoost, respectively. When investigating the incorrect classifications further, we notice that the LOGs were confused with their nearest neighbors. For instance, five elements[5] at LOG 200 were classified as LOG 300, and two elements at LOG 300 were classified as LOG 350.[6] This is mainly because the number of changes modeled to detail the elements further from LOG 200 to 300 does not necessarily increase the shape complexity enough to be differentiated. Moreover, this approach heavily relies on the dataset size (finding similar observations). Thus, increasing the dataset size even more has a potential for improving the accuracy of the classifications.

4.5. Experiment: Performance robustness evaluation

As discussed previously, the ensemble models developed produce their classifications based on the extracted geometric features. The elements dataset presented in this paper was entirely modeled by using Autodesk Revit. Additionally, the Revit API was used to export the triangulated mesh representations that were used to extract the different geometric features. The Revit API provides specialized methods for retrieving the geometric representation of the individual elements.[7],[8] Our implementation was based on an available example code provided by Autodesk,[9] where we used the maximum value for the *LevelofDetail* parameter when generating the triangulated mesh. Considering that every BIM-authoring tool might have its own geometry kernel, which

[5] Fire Mains, Heating Pipe Fittings, Wood Stair, Trefoil Round Arch Window, Ventilation System.
[6] Multilayered Slab (Reinforced Concrete), Escalator (variant 2).

[7] https://www.revitapidocs.com/2015/d8a55a5b-2a69-d5ab-3e1f-6cf1ee43c8ec.htm.
[8] https://thebuildingcoder.typepad.com/blog/2015/04/exporting-3d-element-geometry-to-a-webgl-viewer.html.
[9] https://jeremytammik.github.io/tbc/a/0792_obj_export_v1.htm#6.

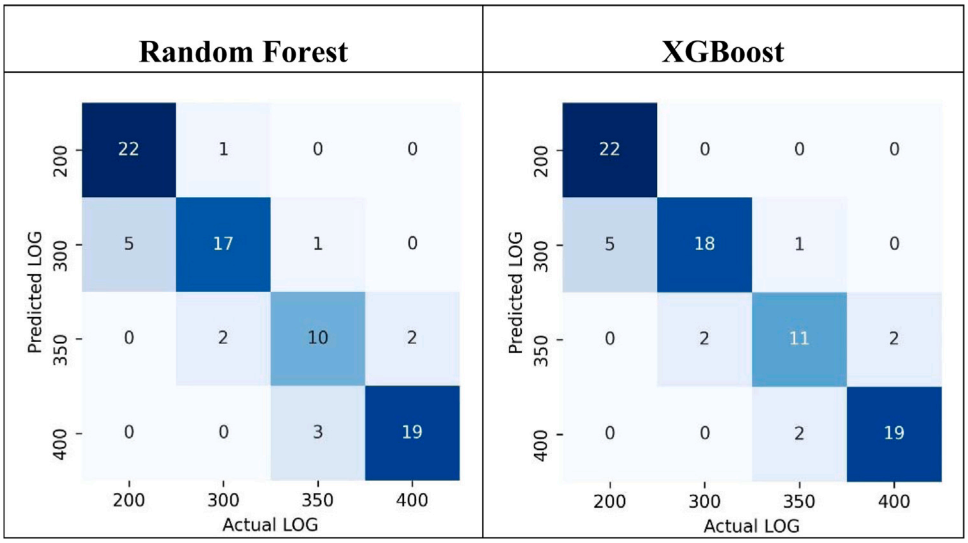
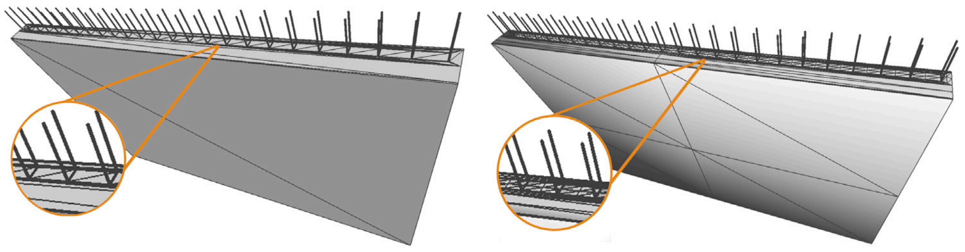


Fig. 15. Evaluation results: confusion matrix of the RF and XGBoost models performance on test data. The *x*-axis represents the actual LOG, and the *y*-axis represents the predicted LOG. The diagonal boxes show the correctly predicted LOG classes.



(a) Before re-meshing: 627,702 vertices, 1,167,543 faces, and 3,589,782 edges.

(b) After re-meshing: 3,189,421 vertices, 6,049,491 faces, and 18,482,220 edges.

Fig. 16. Re-meshing experiment: a sample reinforced wall before and after re-meshing.

could represent the geometry with more or fewer triangles, this experiment aims to evaluate whether the performance of the developed ensemble models would be affected by re-meshing the testing dataset with differently distributed triangles.

The re-meshing process was performed using the Isotropic Explicit re-meshing algorithm [64], where the shapes became more condensed and uniform. The Isotropic Explicit re-meshing can deal with a variety of mesh shapes and is widely used and implemented in multiple geometry processing tools, such as Meshlab.[10] Fig. 16 shows an example of a multilayered reinforced wall before and after re-meshing. The wall's vertices, edges, and faces after re-meshing are ~5.1-fold their values before re-meshing. Any existing BIM-authoring tool will typically export less condensed meshes, which are more similar to the original dataset (before re-meshing). Thus, using such condensed re-meshing is adequate for evaluating the robustness of the trained models' performance.

After re-meshing, the geometric features are extracted again for the re-meshed test dataset. As illustrated by Fig. 16, the basic geometric features of the re-meshed elements are approximately five-fold their values before re-meshing. However, the extracted sharp edges and segments were not affected by re-meshing, as neither the elements' diameter nor their outline has changed. The features that are affected by re-meshing are those which rely on the edges' lengths as well as the count of vertices, faces, and edges. As indicated in Figs. 12 and 14, the affected features are not part of the top four important features. However, the *% of Mean Edges Length* is the fifth feature, and the rest of the features combined also contribute to the final classification of the different LOGs.

After extracting the geometric features for the test dataset, they were used to evaluate the performance of the already trained models (in Sections 4.2 and 4.3). The evaluation results are presented in Table 5. To highlight the difference to the performance before re-meshing (Table 4), the metrics are colored according to the change in their values; green when improved and red when degraded. When comparing the overall accuracy of both models before and after re-meshing, although the RF model metrics have increased and decreased across the different LOGs, it maintained the same accuracy of 83%. In contrast, the accuracy of the XGBoost model dropped from 85% to 78% after re-meshing. The XGBoost has maintained its performance for the LOGs 200 and 300, which is an advantage compared to RF. However, the performance degraded at the LOGs 350 and 400. In this regard, multiple elements at LOG 400 were classified as LOG 350 (see the confusion matrix shown in Fig. 17). We observe that mesh density has only a slight impact on the models' performance. Hence, the trained ensemble models are capable of classifying the LOG of elements that are meshed differently than the training data.

5. Case study

This section highlights the applicability of the developed approach in checking the LOG within the established workflows in practice. As illustrated in Fig. 18, the requirements of the delivered BIM models are typically specified in contracts and BIM execution plans. These

[10] https://www.meshlab.net/.

Table 5
Re-meshing experiment results: performance results of the RF and XGBoost models on test data. The performance metrics are described as precision, recall, and F1-Score. Precision describes the model performance in positive predictions while considering false positives. Recall incorporates false negatives instead of false positives, and F1-score provides a balance between precision and recall. The colors highlight the change in values compared to before re-meshing (Table 4); green when improved and red when degraded.

LOG	Precision		Recall		F1-score	
	RF	XGBoost	RF	XGBoost	RF	XGBoost
200	1.00	1.00	0.78	0.81	0.88	0.90
300	0.70	0.75	0.95	0.90	0.81	0.82
350	0.69	0.52	0.79	0.86	0.73	0.65
400	0.94	0.92	0.81	0.57	0.87	0.71
Accuracy					**0.83**	**0.78**
Macro avg.	0.83	0.80	0.83	0.79	0.82	0.77
Weighted avg.	0.86	0.84	0.83	0.78	0.83	0.79

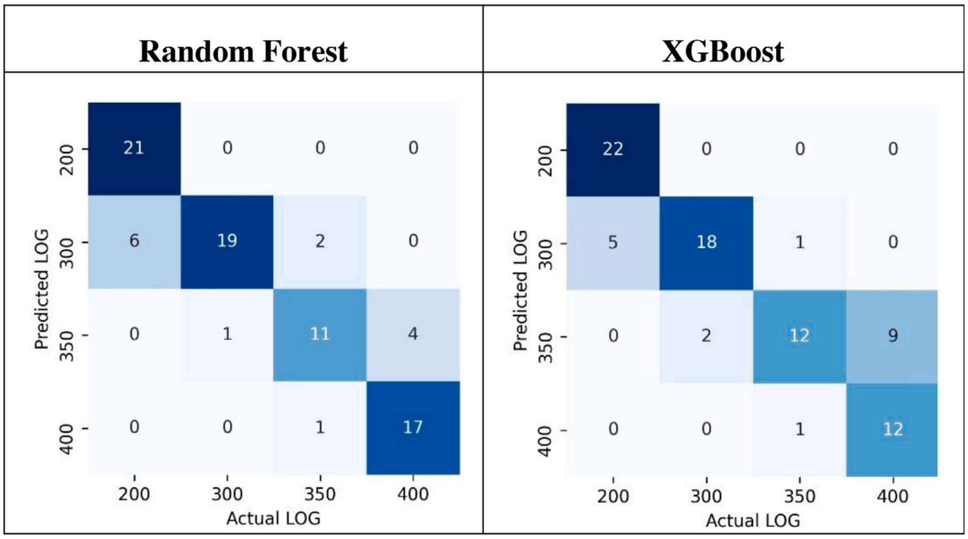


Fig. 17. Re-meshing experiment results: confusion matrix of the RF and XGBoost models performance on test data. The *x*-axis represents the actual LOG, and the *y*-axis represents the predicted LOG. The diagonal boxes show the correctly predicted LOG classes.

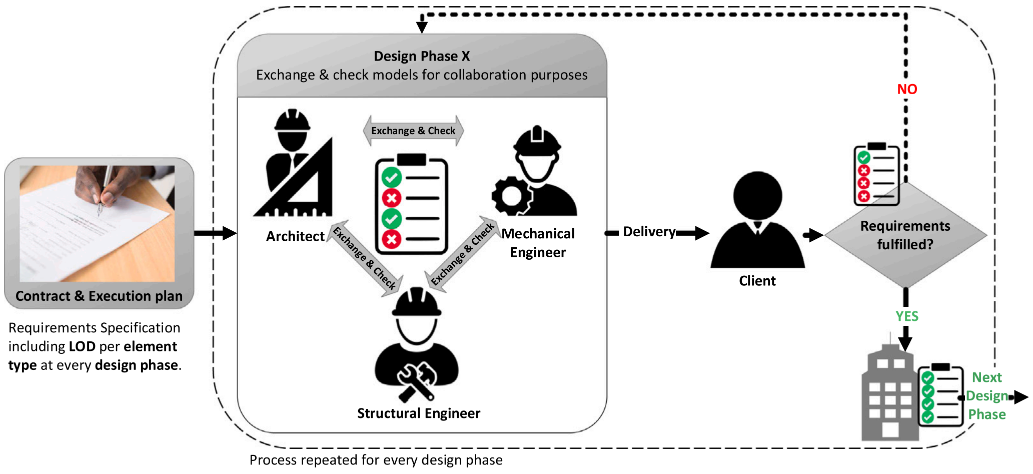

Design Phase X
Exchange & check models for collaboration purposes
Exchange & Check
Architect
Exchange & Check
Exchange & Check
Mechanical Engineer
Structural Engineer
Contract & Execution plan
Requirements Specification including LOD per element type at every design phase.
Delivery
Client
Requirements fulfilled?
NO
YES
Next Design Phase
Process repeated for every design phase

Fig. 18. Illustration of the multidisciplinary design process, highlighting the specification of a project's LOD requirements in contracts and BIM execution plans, and then validating the specified requirements during the collaboration with different disciplines as well as delivery to the client.

specifications include the LOI and LOG of the individual element types required from each domain expert on every design phase.

During every design phase, the different domain experts base their work on models provided by experts from other disciplines. At this point, the exchanged models need to be checked for fulfilling the minimum requirements necessary by the recipient discipline for carrying out its tasks. Once the different models are integrated and handed over to the client, a final quality check for fulfilling the various requirements is performed. When issues are detected in the project delivery, feedback is sent back to the project participants requesting clarification and solving the issues identified. Otherwise, the design phase delivery is confirmed by the client, which will be used as a basis for developing the design further in the subsequent design phase.

Table 6
Example of an LOD specification, showing the required types of building elements and their corresponding LOG and LOI specifications.

Identification		Level of geometry	Level of information (LOI)								
Element type	IFC class	LOG	Name	Core material	Load-bearing function	Surface covering (texture)	Fire protection characteristics	Part of escape route?	Sound insulation characteris-tics	Is external?	Thermal transmittance
Windows	IfcWindow	300	x	x			x	x	x	x	x
Walls	IfcWall	350	x	x	x	x	x	x	x	x	x
Curtain Walls	IfcCurtainWall	350	x	x	x	x	x	x	x	x	x
Stairs	IfcStair	200	x	x				x		x	
Ramps	IfcRamp	200	x	x				x		x	
Doors	IfcDoor	200	x	x				x		x	
Ceiling	IfcCeiling	300	x	x	x		x				
Sanitary	IfcSanitary	100	x								
Rooms	IfcSpace	-	x					x		x	
Slabs	IfcSlab	300	x	x	x		x		x	x	
Roofs	IfcRoof	300	x	x	x		x		x		x
Beams	IfcBeam	200	x	x	x					x	
Columns	IfcColumn	200	x	x	x		x	x		x	
Structural truss	IfcAssembly	200	x	x	x						
Foundation	IfcFooting	350	x	x	x		x				x
Framing	IfcBuildingEle-mentProxy	300	x	x	x		x		x		

The example shown in Table 6 is a subset of the requirements specified for a real-world project (*Ferdinand Tausendpfund GmbH & Co. KG*[11] office building, in Regensburg, Germany). While modeling the conceptual design, the owner decided to build a sustainable building and explore multiple design options through evaluating the performance of their structural system as well as embodied and operational energy consumption [14,45]. The table lists a subset of the mappings between the building element types and their geometric and semantic requirements that must be delivered at the end of the conceptual design.

To emphasize on the integration of checking the LOG within the design process, a plugin that uses the developed approach was developed inside Revit, shown in Fig. 19. The figure shows a design option of the Tausendpfund's office building on the left and the results of checking the LOG of the individual elements on the right. For this purpose, the trained XGBoost model from this study was hosted on a *Flask*[12] server, where the plugin inside Revit sends the geometrical features of the individual elements through a representational state transfer (REST) web-service and receives the predicted LOG as a response. Then each predicted LOG is compared to the project's requirements (which are selected as a CSV file at the top). Finally, the results are reported as lists grouping the elements as *Passed*, *Errors* (when they did not pass), and *Warnings* (for element types that are part of the specification and were not found in the model). Checking the LOG revealed the following deviations:

- Stairs and ramps were identified as LOG 350, whereas they are required to be at LOG 200. The elements used in the model were not as simple as generic representations as they have included detailed railings and connections. After a discussion with the modelers, their reasoning was that the used families are standard and were developed for other similar projects.
- Interior walls were identified as LOG 200 rather than 350. After inspecting the model, we found that the used walls are single-layered walls and do not model the exterior or interior details such as framing, insulation, or connections. Generally, interior walls are not much developed at this phase; however, the specification should have differentiated between interior and exterior walls.
- Entrance door was identified as LOG 300 rather than 200 as the automatic door opener is additionally modeled.

[11] https://www.tausendpfund.group/.
[12] https://flask.palletsprojects.com/en/2.0.x/.

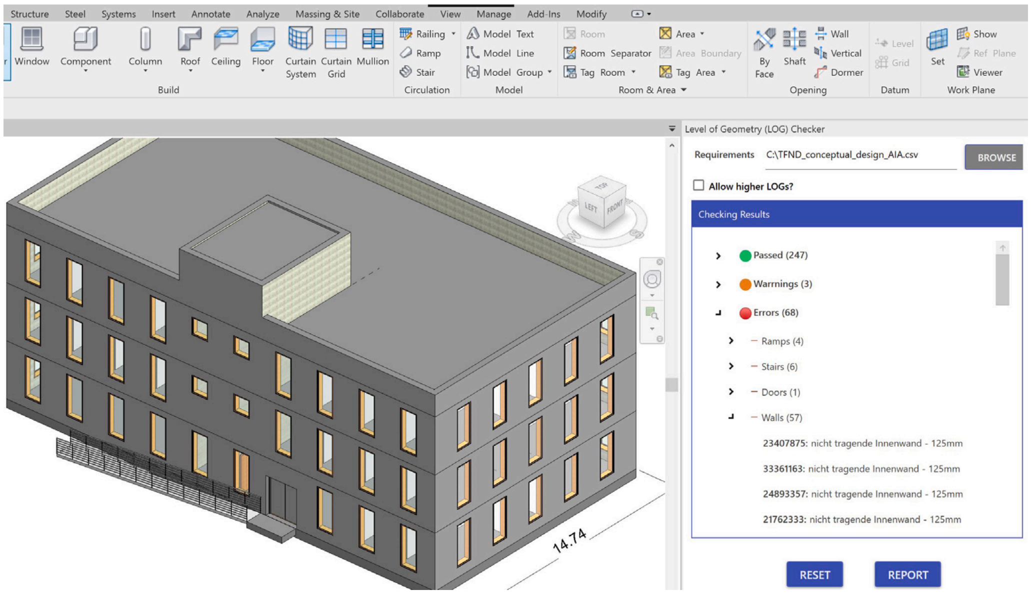
Reconsidering the XGBoost's evaluation matrix (Fig. 15), the model's accuracy is 83% due to confusion with the adjacent LOGs. However, no LOG was confused with another LOG that is higher or lower than one level, like confusing LOG 200 with LOG 350. Hence, in this case study, the trained model could raise certain warning flags when the modeled LOG is not compliant with the specification. In some cases, when there is no considerable increase in detailing between the LOGs 200 and 300, the model's prediction might be less accurate. In those cases, it would be helpful to inform practitioners about the prediction probability (e.g., 65% to highlight any potential inaccuracies) as well as enhance the accuracy of the LOG prediction by checking the provided semantics. Additionally, custom industry cases can be handled with tailored behaviors, such as considering elements at a higher LOG than what is required as compliant and marking them as passed.

6. Conclusions and future research

The automatic validation of building information for compliance with the design requirements is crucial for an efficient and successful project outcome. The LOD concept is used to specify the expected information of the individual elements. Currently, automatically checking the completeness of the semantic information against the LOD specification is supported by multiple tools. However, validating the conformance of the provided geometry to the required LOG is currently a manual, laborious task and solely based on domain experts' subjective assessment.

This paper contributes a framework for formally defining and automatically checking the LOG of building elements. The proposed approach is based on modeling and analyzing a dataset of 408 building elements (102 families at the LOGs 200, 300, 350, and 400). The families were modeled according to the most established and widespread LOD specifications, the BIMForum's LOD specification and Trimble's Project Progression Planning, and are provided to the scientific community as open data. The existing descriptions and especially the graphical illustrations from these specifications were used as baseline for the modeling process. Additionally, measuring the necessary modeling time (which reflects the required effort and cost) to detail the elements further from one LOG to the subsequent one showed a 1.88 – 2.80-fold increase.

From the experience gained in this study, we highlight that even when modelers might have different interpretations of the fine details expected at each LOD, the outcome would be comparable and sufficient


Structure Steel Systems Insert Annotate Analyze Massing & Site Collaborate View Manage Add-Ins Modify
Window Component Column Roof Ceiling Floor Curtain System Curtain Grid Mullion
Build
Railing Ramp Stair
Circulation
Model Text Model Line Model Group
Model
Room Room Separator Tag Room Area Area Boundary Tag Area
Room & Area
By Face Shaft Wall Vertical Dormer
Opening
Level Grid
Datum
Set Show Ref Plane Viewer
Work Plane
14.74

Level of Geometry (LOG) Checker

Requirements C:\TFND_conceptual_design_AIA.csv BROWSE

☐ Allow higher LOGs?

Checking Results

- 🟢 Passed (247)
- 🟠 Warrnings (3)
- 🔴 Errors (68)
  - Ramps (4)
  - Stairs (6)
  - Doors (1)
  - Walls (57)
    - 23407875: nicht tragende Innenwand - 125mm
    - 33361163: nicht tragende Innenwand - 125mm
    - 24893357: nicht tragende Innenwand - 125mm
    - 21762333: nicht tragende Innenwand - 125mm

RESET REPORT

Fig. 19. A snapshot of the developed plugin inside Revit for checking the LOG of building elements.

as long as they follow a unified specification and understand which features must be modeled (such as the exact shape dimensions at LOG 300 or connections at LOG 350). Additionally, following and interpreting the specifications to model the geometry can be improved by providing graphical illustrations for all families. Currently, numerous families are only described with textual descriptions. Furthermore, providing open-access LOG examples would support achieving a common understanding of what needs to be modeled. Currently, no LOG examples are available online (other than the dataset published by this research).

On the basis of the created dataset, the detailing of the individual building elements at each LOG was formally analyzed and represented by a set of features. A main scientific contribution of the paper is the identification of the relevant geometric features. The geometric features were extracted using multiple geometry analysis techniques, including the basic geometric features (such as vertices, faces, and edges), sharp edges, and diameter-based segmentation. The extracted features were then used to train RF and XGBoost models to classify the LOG of any building element.

The results show that the extracted geometric features can describe the elements' complexity in a way that represents the modeled features at every LOG. Both of the ensemble models were able to classify the LOG of the test dataset with an accuracy of 83% for the RF model and 85% for the XGBoost model. After detailed investigations (as shown in Fig. 15), we found that the misclassified elements (18% for RF and 15% for XGBoost) were only confused with their nearest neighbors, e.g., LOG 200 with LOG 300. Hence, both of the trained models are capable of providing practitioners a reliable indicator of the geometric detailing of the individual elements. Additionally, to enhance the reliability of predictions, we propose informing practitioners about the predictions' probability to highlight potential inaccuracies that require a manual inspection.

In order to evaluate the robustness of the trained models, the test dataset was re-meshed in a much denser triangulation, where the RF model maintained its accuracy of 83%. In contrast, the accuracy of the XGBoost model dropped to 78%. These results demonstrate the capability of the trained models in correctly classifying the LOG of elements that are meshed differently than the training data. Accordingly, using the RF model in practice would provide more robust accuracy for classifying the LOG, when the evaluated families are provided from diverse BIM-authoring tools.

In general, the ensemble models have proven their capability of learning the geometric features. Increasing the dataset size further has a potential for improving the ensemble models' accuracy, especially given that the change from one LOD to the other does not substantially increase the shape's complexity for all the families. Based on the knowledge gained from this paper, the process of extracting geometric features is a sensitive and time-consuming task. As shown in this study, the elements' geometry must be pre-processed into a set of (human-made) representative features when using ensemble learners, which involves performing multiple computationally extensive tasks. Therefore, as a next step, mesh convolutional neural networks will be evaluated for directly extracting geometric features and classifying the LOG of triangulated meshes. Classifying the LOG of building elements directly from meshes would reduce the necessary processing effort and could improve accuracy, since the geometric features would be statistically inferred by the neural network rather than manually identified and extracted.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

We gratefully acknowledge the financial support of the German Research Foundation (Deutsche Forschungsgemeinschaft, DFG) in the frame of the programs FOR 2363 "Early BIM" and TRR 277 "Additive Manufacturing in Construction – The Challenge of Large Scale" (project no. 414265976). We thank Ferdinand Tausendpfund GmbH for providing their office building as a sample project. We also thank Patrick Nordmann for his contribution in modeling the LOG dataset.

References

[1] I. Howell, The value information has on decision-making, New Hamps. Bus. Rev. 38 (2016) 19.

[2] M. Hooper, Automated model progression scheduling using level of development, Constr. Innov. 15 (4) (2015) 428–448.

[3] F. Steinmann, Modellbildung und computergestütztes Modellieren in frühen Phasen des architektonischen Entwurfs, (Ph.D. thesis), Bauhaus University, Weimar, Germany, 1997.

[4] J. Beetz, A. Borrmann, M. Weise, Process-based definition of model content, in: Building Information Modeling, Springer, 2018, pp. 127–138.

[5] ISO, ISO 9000:2015(en) Quality management systems — Fundamentals and vocabulary, 2015, URL https://www.iso.org/obp/ui/es/#iso:std:iso:9000:ed-4:v1:en, (visited 2020-04-01).
[6] BIMForum, Level of development specification guide, 2019, URL http://bimforum.org/lod/, (visited 2020-03-19).
[7] F. Leite, A. Akcamete, B. Akinci, G. Atasoy, S. Kiziltas, Analysis of modeling effort and impact of different levels of detail in building information models, Autom. Constr. 20 (5) (2011) 601–609.
[8] L. van Berlo, F. Bomhof, Creating the dutch national BIM levels of development, in: Computing in Civil and Building Engineering (2014), 2014, pp. 129–136.
[9] M. Bolpagni, A.L.C. Ciribini, The information modeling and the progression of data-driven projects, in: CIB World Building Congress, 2016, pp. 296–307.
[10] Trimble, Project progression planning with MPS 3.0, 2013, URL http://support.vicosoftware.com/FlareFiles/Content/KB/Trimble%20-%20Progression%20Planning%20V15.pdf, (visited 2020-03-19).
[11] J. Abualdenien, A. Borrmann, Vagueness visualization in building models across different design stages, Adv. Eng. Inf. 45 (2020) 101107, URL http://www.sciencedirect.com/science/article/pii/S1474034620300768.
[12] P. Schneider-Marin, H. Harter, K. Tkachuk, W. Lang, Uncertainty analysis of embedded energy and greenhouse gas emissions using BIM in early design stages, Sustainability 12 (7) (2020) 2633.
[13] A. Zahedi, J. Abualdenien, F. Petzold, A. Borrmann, Minimized communication protocol based on a multi-LOD meta-model for adaptive detailing of BIM models, in: Proc. of the 26th International Workshop on Intelligent Computing in Engineering, Leuven, Belgium, 2019.
[14] J. Abualdenien, A. Borrmann, A meta-model approach for formal specification and consistent management of multi-LOD building models, Adv. Eng. Inf. 40 (2019) 135–153.
[15] A. Géron, Hands-on Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques To Build Intelligent Systems, O'Reilly Media, 2019.
[16] J. Zhang, G. Ma, Y. Huang, J. sun, F. Aslani, B. Nener, Modelling uniaxial compressive strength of lightweight self-compacting concrete using random forest regression, Constr. Build. Mater. 210 (2019) 713–719, URL http://www.sciencedirect.com/science/article/pii/S0950061819306798.
[17] T. Chen, C. Guestrin, Xgboost: A scalable tree boosting system, in: Proceedings of the 22nd Acm Sigkdd International Conference on Knowledge Discovery and Data Mining, 2016, pp. 785–794.
[18] W. Dong, Y. Huang, B. Lehane, G. Ma, Xgboost algorithm-based prediction of concrete electrical resistivity for structural health monitoring, Autom. Constr. 114 (2020) 103155, URL http://www.sciencedirect.com/science/article/pii/S0926580519311148.
[19] A. Braun, S. Tuttas, A. Borrmann, U. Stilla, Improving progress monitoring by fusing point clouds, semantic data and computer vision, Autom. Constr. 116 (2020) 103210, URL http://www.sciencedirect.com/science/article/pii/S0926580519309975.
[20] A. Borrmann, M. König, C. Koch, J. Beetz, Building Information Modeling Technology Foundations and Industry Practice: Technology Foundations and Industry Practice, Springer, 2018.
[21] A. Borrmann, V. Berkhahn, Principles of geometric modeling, in: A. Borrmann, M. König, C. Koch, J. Beetz (Eds.), Building Information Modeling: Technology Foundations and Industry Practice, Springer International Publishing, Cham, 2018, pp. 27–41.
[22] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, B. Lévy, Polygon Mesh Processing, CRC Press, 2010.
[23] M. Garland, Quadric-Based Polygonal Surface Simplification, Carnegie Mellon University, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science, 1999.
[24] D. Shikhare, Complexity of Geometric Models, National Centre for Software Technology, Mumbai, 2001.
[25] L. Shapira, A. Shamir, D. Cohen-Or, Consistent mesh partitioning and skeletonisation using the shape diameter function, Vis. Comput. 24 (4) (2008) 249.
[26] R. Hanocka, A. Hertz, N. Fish, R. Giryes, S. Fleishman, D. Cohen-Or, MeshCNN: a network with an edge, ACM Trans. Graph. 38 (4) (2019) 1–12.
[27] R. Nikhila, R. Jeremy, N. David, G. Taylor, W. Lo, J. Justin, G. Georgia, Accelerating 3d deep learning with pytorch3d, 2020, arXiv:2007.08501.
[28] A.R. Forrest, Computational geometry-achievements and problems, in: Computer Aided Geometric Design, Elsevier, 1974, pp. 17–44.
[29] L. Heisserman, Generative geometric design, IEEE Comput. Graph. Appl. 14 (2) (1994) 37–45.
[30] J. Abualdenien, A. Borrmann, Formal analysis and validation of Levels of Geometry (LOG) in building information models, in: Proc. of the 27th International Workshop on Intelligent Computing in Engineering, Berlin, Germany, 2020.
[31] D. Luebke, M. Reddy, J. Cohen, A. Varshney, B. Watson, R. Huebner, Level of Detail for 3D Graphics, Morgan Kaufmann, 2003.
[32] T. Kolbe, G. Gröger, L. Plümer, Citygml: Interoperable access to 3D city models, in: Geo-Information for Disaster Management, Springer, 2005, pp. 883–899.
[33] VicoSoftware, Bim level of detail, 2005, URL http://www.vicosoftware.com, (visited 2020-05-13).
[34] AIA, Building information modeling protocol exhibit, 2008, URL https://bit.ly/2NZBbTV, (visited 2020-05-07).
[35] BSI, Pas 1192-2:2013 specification for information management for the capital/delivery phase of construction projects using building information modelling, 2017, URL http://shop.bsigroup.com/Navigate-by/PAS/PAS-1192-22013/, (visited on 2020-01-11).
[36] VBI, BIM-Leitfaden für die Planerpraxis, 2016, URL https://www.vbi.de/fileadmin/redaktion/Dokumente/Infopool/Downloads/VBI_BIM-Leitfaden_0916-final.pdf, Verband Beratender Ingenieure (VBI), https://www.vbi.de (visited on 2020-05-08).
[37] PROGETTIAMOBIM, I gradi dei lod. Le UNI 11337-4 confrontate con la scala Americana, 2018, URL https://www.progettiamobim.com/blog/approfondimenti/i-gradi-dei-lod/, (visited on 2020-06-11).
[38] C. Maier, Grundzüge einer open BIM Methodik für die Schweiz - Version 1.0, 2015, URL https://www.ebp.ch/sites/default/files/2016-11/ki-leitfaden-open-bim.pdf, (visited on 2020-05-29).
[39] DIN, DIN EN 17412:2019-07 building information modelling - level of information need - concepts and principles, 2019, URL https://www.beuth.de/en/draft-standard/din-en-17412/306353980, (visited 2020-04-01).
[40] A. Gigante-Barrera, D. Ruikar, S. Sharifi, K. Ruikar, A grounded theory based framework for level of development implementation within the information delivery manual, Int. J. Inf. Model. (IJ3DIM) 7 (1) (2018) 30–48.
[41] P. Schneider-Marin, J. Abualdenien, A framework to facilitate an interdisciplinary design process using BIM, in: Proc. of the 31th Forum Bauinformatik, Berlin, Germany, 2019.
[42] S. Vilgertshofer, A. Borrmann, Using graph rewriting methods for the semi-automatic generation of parametric infrastructure models, Adv. Eng. Inf. 33 (2017) 502–515.
[43] H. Abou-Ibrahim, F. Hamzeh, Enabling lean design management: An LOD based framework, Lean Constr. J. 2016 (2016) 12–24.
[44] I. Grytting, F. Svalestuen, J. Lohne, H. Sommerseth, S. Augdal, O. Lædre, Use of LoD decision plan in BIM-projects, Procedia Eng. 196 (2017) 407–414.
[45] J. Abualdenien, P. Schneider-Marin, A. Zahedi, H. Harter, H. Exner, D. Steiner, M. Mahan Singh, A. Borrmann, W. Lang, F. Petzold, M. König, P. Geyer, M. Schnellenbach-Held, Consistent management and evaluation of building models in the early design stages, J. Inf. Technol. Constr. 25 (2020) 212–232.
[46] H. Exner, J. Abualdenien, M. König, A. Borrmann, Managing building design variants at multiple development levels, in: Proc. of the 36th International Council for Research and Innovation in Building and Construction (CIB W78), Newcastle, UK, 2019.
[47] K. Wong, C. Ellul, Using geometry-based metrics as part of fitness-for-purpose evaluations of 3D city models, in: ISPRS Annals, Vol. 4, 2016, pp. 129–136.
[48] J.R. Quinlan, Bagging, boosting, and C4. 5, in: AAAI/IAAI, Vol. 1, 1996, pp. 725–730.
[49] L. Breiman, Random forests, Mach. Learn. 45 (1) (2001) 5–32.
[50] L. Raileanu, K. Stoffel, Theoretical comparison between the gini index and information gain criteria, Ann. Math. Artif. Intell. 41 (1) (2004) 77–93.
[51] L. Breiman, Bagging predictors, Mach. Learn. 24 (2) (1996) 123–140.
[52] J.R. Quinlan, Learning decision tree classifiers, ACM Comput. Surv. 28 (1) (1996) 71–72.
[53] Y. Freund, R.E. Schapire, A desicion-theoretic generalization of on-line learning and an application to boosting, in: P. Vitányi (Ed.), Computational Learning Theory, Springer Berlin Heidelberg, Berlin, Heidelberg, 1995, pp. 23–37.
[54] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, T.-Y. Liu, Lightgbm: A highly efficient gradient boosting decision tree, in: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (Eds.), Advances in Neural Information Processing Systems 30, Curran Associates, Inc, 2017, pp. 3146–3154, URL http://papers.nips.cc/paper/6907-lightgbm-a-highly-efficient-gradient-boosting-decision-tree.pdf.
[55] D. Zhang, A. Johnson, M. Hebert, Y. Liu, On generating multi-resolution representations of polygonal meshes, 1998.
[56] K. Hildebrandt, K. Polthier, M. Wardetzky, Smooth feature lines on surface meshes, in: Symposium on Geometry Processing, 2005, pp. 85–90.
[57] O. Monga, N. Armande, P. Montesinos, Thin nets and crest lines: Application to satellite data and medical images, in: Proceedings., International Conference on Image Processing, Vol. 2, 1995, pp. 468–471.
[58] C. Weber, S. Hahmann, H. Hagen, Sharp feature detection in point clouds, in: 2010 Shape Modeling International Conference, 2010, pp. 175–186.
[59] K. Pearson, VII. Mathematical contributions to the theory of evolution.—III. Regression, heredity, and panmixia, Philos. Trans. R. Soc. Lond. Ser. A (187) (1896) 253–318.
[60] openml.org, 10-Fold crossvalidation, 2020, URL https://www.openml.org/a/estimation-procedures/1, (visited on 2020-11-01).
[61] S.M. Lundberg, S.-I. Lee, A unified approach to interpreting model predictions, in: Advances in Neural Information Processing Systems, 2017, pp. 4765–4774.
[62] E. Štrumbelj, I. Kononenko, Explaining prediction models and individual predictions with feature contributions, Knowl. Inf. Syst. 41 (3) (2014) 647–665.
[63] M.T. Ribeiro, S. Singh, C. Guestrin, "Why should I trust you?" Explaining the predictions of any classifier, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 1135–1144.
[64] P. Alliez, E.C. De Verdire, O. Devillers, M. Isenburg, Isotropic surface remeshing, in: 2003 Shape Modeling International, IEEE, 2003, pp. 49–58.