The published version is found in the ASCE Library here: https://ascelibrary.org/doi/10.1061/%28ASCE%29CP.1943-5487.0000858 Wu, J., and Zhang, J. (2019). "New Automated BIM Object Classification Method to Support BIM Interoperability." Journal of Computing in Civil Engineering, 33(5), 04019033.

**CP2922**

# New Automated BIM Object Classification Method to Support BIM Interoperability

Jin Wu, S.M.ASCE[1]; and Jiansong Zhang, A.M.ASCE[2]

[1] Graduate Student, School of Construction Management Technology, Purdue University, 401 N Grant Street, West Lafayette, IN 47907.
[2] Assistant Professor, School of Construction Management Technology, Purdue University, 401 N Grant Street, West Lafayette, IN 47907 (corresponding author). E-mail:zhan3062@purdue.edu; Tel: +1-765-494-1574; Fax: +1-765-496-2246.



## Abstract

Industry foundation classes (IFC) is widely accepted as the future of building information modeling (BIM) to take on the challenge of BIM interoperability and enables its support of various automation tasks. However, it is not uncommon to see misuses of IFC entities during the creation of BIM. Such misuses prevent a successful automation of BIM-supported tasks because misclassification of objects in BIM can lead to significant negative consequences in downstream applications due to incorrect semantic information provided. To address this problem, the authors propose a new data-driven, iterative method that can be used to develop an algorithm to automatically classify each object in an IFC model into predefined categories. The algorithm consists of multiple sub-algorithms with each sub-algorithm depicting a pattern matching rule that uses inherent features of the geometric representation of an architecture, engineering, and construction (AEC) object. The method was tested in an experiment where IFC models from three different sources were collected and 1,891 AEC objects were extracted and divided into training and testing data for use. By comparing the classification results of the algorithm developed based on training data and applied to testing data with a manually developed gold standard, 84.45% recall and 85.20% precision were achieved in common building element categories, 100% recall and

precision were achieved in detailed beam categories. The sources of errors were found to be: (1) different objects sharing the same geometric shape; and (2) uncovered geometric shape representation in the training data. By adding locational information into consideration in addition to geometric information and making sure training data covers all geometric shape representations, 100% precision and recall can be achieved for all categories.

**Author keywords:** Object classification; Building information modeling (BIM); Interoperability; IFC; Computer applications.

## Introduction

Since the emergence of the first Building information modeling (BIM) software in the 1980s - the ArchiCAD's Radar CH - BIM has been under fast development due to its high demand (Quirk 2012). BIM software, such as AutoDesk Revit, Bentley AECOsim, Solibri Model Viewer, and BIMserver, help people visualize a building design prior to its construction (Eastman et al. 2011). Such pre-construction visualization improves collaboration between different stakeholders such as planner, designer, structural engineer, construction manager, and field workers. A better collaboration between these stakeholders will likely improve performance and quality of the end product – the building. With the BIM software at hand, people can take quick responsive actions to design changes and discover errors and omissions prior to the actual construction (Eastman et al. 2011). There are many off-the-shelf BIM software that a BIM user can choose from. For example, ETakeoff helps people with the cost estimation task, ANSYS helps people with the structural analysis task, and STR Vision provides a 4D and 5D modeling system to help people with scheduling and cost estimation tasks. For such BIM uses, one would like to take advantage of all the strengths of these different BIM software. Considering that for the same project the BIM data in these different software belong to the same building structure, it will be a waste of

opportunities to create BIM data from scratch in each of the software. An optimal process would require a seamless data transfer between different software in an automated fashion. In theory, BIM is designed to be interoperable. However, in practice, BIM software developed by different companies use different data structures and data formats. As a result, data transfer between different software used by stakeholders in different disciplines can be costly because of the needed manual efforts or converters in conducting the transfer and fixing information inconsistency or adding missing information, even for the same building project. For example, if two software need one converter that converts the BIM model between the formats of the two software, ten BIM software would require $C_{10}^{2} = 45$ converters (theoretically) to achieve interoperability between all of them. The number of converters increases quadratically with the number of BIM software, resulting in a high cost and inconvenience for achieving interoperability between all BIM software. To reduce the high cost of achieving BIM interoperability, twelve companies collaborated to develop a uniform standard for BIM which is known as the Industry Foundation Classes (IFC) (Hamil 2012).

IFC specifications are open and transparent (buildingSMART 2018a). IFC models can be readily accessed using a text editor. Under constant development and refinement by buildingSMART, IFC became one of the most promising attempts trying to solve BIM interoperability. However, IFC schema still has major problems when transforming to other proprietary software formats and vice versa (Ma et al. 2006; Pazlar and Turk 2008). The IFC standard defines its own object data types to store the physical and functional information of building elements in entities and attributes. For a standard building, most elements are in common shapes such as cuboids, prisms, and cylinders, whereas uncommon shapes (in the context of building structure) also exist such as pyramids, pentagonal cylinders, and dodecahedrons. It is easier to construct elements in common shapes than

the ones in uncommon shapes. However, it also creates room for the misuse of entities. For example, an *IfcSlab* entity may be misused to represent a wall, which should be using *IfcWall* or *IfcWallStandardCase* for data representation. The misuse occurred because both of them have a cuboid shape, and the only differences are in their corresponding length/width/height ratios. Although model visualization tools can still provide the same visualization results of a slab represented using an *IfcWall* entity as if it was represented correctly by an *IfcSlab* entity, this type of misuse can lead to problems in transferring data between different BIM applications that require the use of semantic information of building elements beyond their shape and geometry, such as architectural design, cost estimation, and structural analysis. It will make the conversion results from IFC files error-prone. For example, given the previous misuse case, when taking off the volume of slabs, the software will mistakenly add the volume of the wall represented by a slab entity. While the names of IFC entities can be misleading in the case of entity misuse, the geometric shapes information are one of the most reliable parts of BIM and it is almost always accurate for a successful building object representation. In this paper, the authors leverage the geometric information of objects in IFC models to automatically classify BIM objects in the architectural, engineering, and construction (AEC) domain into predefined building element categories.

## Background

### *Object Classification*

Object classification is the process where objects are recognized, differentiated, and understood, meaning that the objects are grouped for some specific purpose (Henri et al. 2005). Object classification in an automated fashion involves two essential steps: feature extraction and feature-based classification (Ullman 2007).

There are many researches on 2D object classification to detect and classify objects from 2D images. For example, Henri et al. (2017) proposed to extract a hierarchy of fragments with visual element features such as human faces and car wheels for use in recognizing and classifying objects (e.g., people and cars) from 2D images. Ullman and Epshtein (2007) proposed two extensions of the fragment-based object recognition scheme. One is a hierarchical decomposition into parts and sub-parts at multiple levels according to the features. The other is depicting different views of the same object part using semantically equivalent feature sets (Ullman and Epshtein 2007). Distinguished from the fragment-based scheme, Wang et al. (2009) designed a technique that sorts objects into predefined categories by building their "text-based image features". A text-based feature is built from the tags of its *k*-nearest neighbors in the training collection. For example, "motorbikes" is a text-based feature built from the tags of its 5,000-nearest neighbors based on measuring the chi-square distance in a space of 256-dimensional vectors. Wang et al. (2009) found that text-based features from images are reliable to classify the objects in images even when an object appearance changes in the images. In the civil engineering domain, computing methods and algorithms have also been developed to identify/classify the following contents from 2D images: construction equipment (Memarzadeh et al. 2013), construction activities (Liu and Golparvar-Fard 2015), safety harness (Fang et al. 2018), construction materials (Han and Golparvar-Fard 2015), highway assets (Golparvar-Fard et al. 2013) and traffic signs (Balali and Golparvar-Fard 2015), bridge components (Narazaki et al. 2018), cracks and defects in a structure (Feng et al. 2017; Gopalakrishnan et al. 2017), among others.

In contrast to 2D object classification, 3D object classification has more information to leverage because of its additional spatial dimension. For a successful 3D object classification, object detection is a critical step when dealing with less structured data such as point cloud data collected

using Light Detection and Ranging (LIDAR) techniques. For example, Wang and Schenk (2010) designed an object classification technique to detect and reconstruct buildings from LIDAR data. Their approach uses LIDAR terrain surface, edges, and points of a building as features. Also working on point cloud data but from a different perspective, Voegtle and Steinle (2003) designed a method to detect segments and extract objects inside these segments based on a special region growing algorithm. LIDAR data has been widely used in the civil engineering domain for capturing as-built projects (Wang and Cho 2014), prefabricated components (Kalasapudi et al. 2015), construction equipment and assets (Chen et al. 2016; Fang et al. 2016), and surveying results (Tang and Akinci 2012). RGB-D data is another type of commonly used 3D data other than point cloud data. Different methods have been proposed to conduct object classification on RGB-D data. For example, Richard et al. (2012) proposed a combination of convolutional and recursive neural networks (CNN and RNN) to classify 3D objects from RGB-D data, where multiple RNN weights are randomly initialized and a tree structure is built for the classifier. Bo et al. (2013) proposed a hierarchical matching pursuit (HMP) method for RGB-D data object classification, which uses an unsupervised learning technique with sparse coding to generate hierarchical feature representations for classifying household objects.

In spite of the different types of data used, the above methods all focused on object classification in as-built models or assets. As-built models are models created from surveying/inspecting a physical system (Hefele and Dolin 1998). On the contrary, as-designed models are virtual models that are derived from design data (Huber et al. 2011). As such, as-designed model has more "degrees of freedom" in terms of the possible model setup. For example, an as-designed model can be built as high as the designer wants, whereas as-built models can only be as high as it physically stands. As a result, a successful classification of objects in an as-designed model will

heavily rely on the correct understanding of the designed structure and the BIM data structure. Object classification of as-designed models is underexplored comparing to that of as-built models. In this domain, Qin et al. (2014) proposed a 3D CAD object classification method using deep neural networks in which they leveraged prior CAD knowledge to generate features to use in the deep neural network model training. An average correct rate of 98.64% was achieved in a dataset that contained objects in 28 categories such as screw and nut. Henn et al. (2012) presented a support vector machines (SVMs)-based machine learning classifier that can automatically classify the building type of a selected 3D model from city models such as terraced building and apartment building. They achieved a cross validation accuracy of 90.79% on 1,953 building objects.

***IFC Schema***

IFC schema has been under constant development and released many versions from IFC1.0 to IFC4 Add2 (buildingSMART 2018a). Although not the latest version, IFC2x3 is currently the most widely implemented version of the IFC schema, which the authors chose to use in this paper. In IFC2x3, there are 117 defined types and 653 entities, including 33 types of entities for building elements (buildingSMART 2007). For example, *IfcBeam* is a building element entity that is used to define a beam instance, and *IfcDoor* is a building element entity that is used to define a door instance. Fig. 1 provides a visualization of a beam in IFC and Fig. 2 shows the IFC data of the beam. It is a wide flange beam (or I-beam) that is commonly seen in a steel structure. The *IfcBeam* references other entities such as *IfcOwnerHistoty*, *IfcLocalPlacement*, and *IfcProductDefinitionShape*, which contain detailed information about the beam. By tracing the references between entities in an IFC data, all relevant information to an object can be extracted (Won et al. 2013).

An IFC model of a building usually contains multiple building elements. While most of the building elements should be using correct IFC entities, e.g., *IfcWall* for a wall, *IfcSlab* for a Slab, it is not rare to see misuses, such as the misuse of an *IfcSlab* for a wall as described above. In extreme cases, most of the objects in a model contain entity misuses. For example, in the 59 objects in a bridge model described by Ma et al. (2017), 35 objects (59%) contained entity misuses.

To prevent potential negative consequences resulting from misuses of IFC entities and support a seamless interoperability of BIM, the authors propose an IFC-based BIM object classification method based solely on the geometric information of the object.

***IFC Object Classification***

Few recent works in IFC object classification were found. For example, Koo and Shin (2018) explored the use of novelty detection machine learning approach to detect IFC objects misclassifications during manual creation of the IFC data. They used SVMs to classify objects into individual classes and tested the SVMs on four classes - walls, doors, columns, and slabs. Their testing achieved an accuracy that ranged from 80.95% to 97.14% for different classes. The use of machine learning was promising, but it was difficult (if not impossible) to achieve 100% accuracy. In contrast, Sacks et al. (2017) captured domain expert knowledge into computer rules for classifying IFC objects. The rules were based on pairwise geometric, spatial, and topological relationships between IFC objects. Ma et al. (2017) designed a similar method to classify BIM objects using a tailored matching algorithm. In their methods, each object in consideration is paired with all other objects and the similarity of objects in each pair is calculated by comparing their feature values and relationships. Perfect testing results (100% accuracy) have been reported in both methods on 333 objects and 390 objects from one and two bridge IFC models, respectively. However, their methods focused on the relative relationship between features of different objects

rather than the feature values of the objects themselves, such as geometric representation and numerical parameters of the object, thus, reference objects are always in need. In this paper, the authors look into the geometric representations of objects in IFC and directly leverage such information in classifying the objects. Using the authors' method, although sample objects are still needed in the development phase, the developed algorithm could then be directly applied to new IFC objects, without the need of reference objects during the classification application stage. For example, a regular wall usually has a long rectangular box shape defined by the three parameters of length, width, and height. These geometric parameters can be directly used for BIM object classification. Based on this idea, the authors propose a data-driven method that could be used to develop algorithms for classifying BIM objects automatically. The method was implemented on processing IFC data, but is adaptable to any data format.

**Proposed Method for Automated Object Classification in IFC-based BIM**

The authors propose a new 7-step iterative method to classify BIM objects in IFC models (Fig. 3): data collection, preprocessing, environment setup, primary development, secondary development, error analysis and training improvement, and testing. The method provides a platform for algorithm development using a data-driven and pattern matching rule-based approach. With the addition of detailed patterns, the algorithm can be continuously developed to reach a required level of granularity, e.g., to distinguish beams from walls, to distinguish I-Beams from C-beams, or to distinguish different sizes of I-Beams. The detailed explanation of the 7 steps is listed below. To help illustrate the method, some implementation examples are used in this explanation.

***(1) Data collection: collect IFC models from different sources to create a dataset with a broad coverage of different types of IFC entity usage.***

Although IFC was designed to be an open and neutral data standard that is intended to be used by all disciplines and all life cycle phases of a project in the AEC domain, its built-in flexibility allows the IFC standard to be used in different ways. For example, the same 3D shape can be represented using either a "Swept Solid" (i.e., the solid created by the sweeping motion of an existing solid or plane) or a "Boundary Representation" (i.e., a solid created by a collection of connected surface elements). Furthermore, the existence of property sets allows BIM implementations to customize and define their own properties. Therefore, a dataset consisted of models collected from different sources are expected to have a broader coverage of different types of representations and uses of IFC entities comparing to models collected from a single source.

***(2) Preprocessing: extract IFC objects from the collected models, manually label the data, and divide the objects into training set and testing set.***

In order to classify the objects of an IFC model, the algorithm needs to detect them and extract all related information. The extraction of IFC objects is achieved using the algorithm of Won et al. (2013) as reproduced by the authors that can extract all building elements from an IFC file and store each element as a separate file. Each file contains a building element that is independent of other parts of the original IFC model. For example, one file may contain a window of an exterior wall, whereas another file may contain a slab on the second floor. Fig. 4 shows the visualization of a Duplex Apartment model collected from buildingSMARTalliance of the National Institute of Building Sciences (East 2013). Fig. 5 shows two extracted objects from this model. All such objects from the collected data are extracted in this step. The objects are manually labeled with their correct categories by observing each object in a BIM visualization and data display utility.

Labels include two types: existing categories (represented by IFC entity names) in the IFC schema, and non-IFC categories. The existing IFC categories represent common building elements whereas non-IFC categories can define building elements to any level of detail. During the labeling using existing IFC categories, misuse of IFC entities in the collected data will be identified. The division of the IFC objects into a training dataset and a testing dataset is conducted using a data dividing Java program that the authors wrote. The program randomly picks objects from the extracted set and puts them into training or testing set based on a predefined ratio between training and testing data. A common training/testing data ratio to use for statistical learning is 70% to 30% (Kemal and Salih 2007). However, the authors' rule-based learning has more rationality (i.e., based on geometric theorems) built into the training process and therefore requires less training data comparing to statistical learning, in spite of its dependency on the variety of data representations and their distributions. To study such a learning effect, the authors propose a learning curve measure which will be described in detail in the Experiment Section.

***(3) Environment setup: initially build a classification algorithm with no rules or patterns.***

The authors use Java as their developing language because Java provides a convenient platform with a rich set of existing utilities such as java toolboxes of IFC, which provides facilities to extract information from an IFC model. The algorithm to be developed will take a single IFC object file as input and output the category it belongs to. The classification algorithm is initialized to be empty, i.e., with no rules or patterns. This step establishes an environment in which the IFC object classification algorithm and sub-algorithms can be developed. By default, the algorithm classifies an IFC object into an "unknown" category as no pattern matching rules are applicable. In the development stage, the algorithm is developed by extending it with sub-algorithms, e.g., sub-algorithms to classify an object into beams, walls, columns, etc., or to classify a beam into I-beam,

C-beam, rectangular beam, etc. Each sub-algorithm consists of one or more pattern matching-based rules. A pattern matching-based rule defines a pattern consisted of features that could uniquely recognize a category. These features are inherent properties of the AEC objects such as number of sub-components, number of faces, cross section profile, extrusion direction, dimensional ratio, number of straight lines and curves, line connection angle, length, and turn direction. Extraction of these features are achieved using the authors' developed object analysis algorithms similar to the object extraction algorithms. At this step, there are no sub-algorithms or rules.

***(4) Primary development: study the representations of the training set objects in IFC, build rules and develop sub-algorithms to classify objects into existing categories in IFC.***

Existing categories in IFC represent a common and essential set of elements in a building. For example, *IfcBeam*, *IfcColumn*, *IfcFooting*, *IfcSlab*, and *IfcWall* are used to represent beams, columns, footings, slabs, and walls, respectively. However, misuse of IFC categories could happen. For example, a wall should be represented in an IFC model using *IfcWall* or *IfcWallStandardCase*, but it may be represented using any of the other four IFC entities: *IfBeam*, *IfcColumn*, *IfcFooting*, and *IfcSlab*. This may appear to be correct in visualization, but the semantic information carried would be incorrect and therefore cause errors in BIM applications that rely on such semantic information. In this step, training data will be used to develop pattern matching rules to classify the IFC objects into existing IFC categories such as beams, columns, footings, slabs, and walls, based on the geometric representations of the objects. An object in IFC usually has multiple geometric representations for its "Body" and "Axis" (Geiger et al. 2014). The proposed method here focuses on analyzing the "Body" representation, which could be using one of the three major types of solid representation: "Swept Solid," "Boolean Results," and "Brep Bodies"

(buildingSMART 2007; Zhang 2018). The primary development of sub-algorithms follows an iterative process (Fig. 6): (a) Input reading: get an object instance from the training data; (b) Sub-algorithms development: study the "Body" representation of the object instance and develop sub-algorithms to capture the essential features of the "Body" representation for classifying it into the labeled categories in Step (2); (c) Intermediate testing: apply the cumulative sub-algorithms developed up to this point to all the object instances in the training data; (d) Object instances identification: identify the object instances that were either correctly classified, incorrectly classified, or not classified; (e) Results recording; (f) Recursion: get the next object instance from the training data that was not classified, repeat the process until all object instances in the training data are classified.

***(5) Secondary development: study the representations of IFC objects and develop sub-algorithms to classify them into non-IFC defined categories.***

This step aims to further classify the IFC objects into categories that do not have matching IFC entity names in the IFC schema. These are categories that usually define more detailed characteristics of an object but could be defining an object in any dimension. Those objects are expected to be distinguishable based on their geometric information. To identify these object types, the same iterative method as in Step (4) will be used. Each of developed sub-algorithm will be used to identify one specific object type such as I-beam, C-beam, and rectangular beam.

***(6) Error analysis and training improvement: analyze errors in the classification results on the training set, further add/revise sub-algorithms and rules to improve the training performance.***

After the development in Step (4) and Step (5), the algorithm should be able to classify all the objects in the training data. To verify the correctness, the classification results are compared with the manually labeled categories. For the instances with incorrect classification, an error analysis

will be conducted. The error analysis and training improvement step follows a six-step methodology (Fig. 7): (1) Input reading: get an object instance that was classified incorrectly; (2) Rule analysis: analyze the application of sub-algorithms on this instance and find the pattern-based rule that fires on this instance; (3) Rule modification: modify the identified rule to correct the error instance and update the corresponding algorithm; (4) Modification testing: reapply the updated set of sub-algorithms on the training set; (5) Modification updating: if the performance on the training set improves, then accept the update, otherwise decline the update; (6) Recursion: get the next object instance that was classified incorrectly and repeat the procedure until all error instances were tried.

***(7) Testing: apply the developed classification algorithm to testing data for evaluation.***

This is the evaluation section of the method measured by recall and precision. The authors adapted the measurements of recall and precision from information science domain (Makhoul et al. 1999). Recall is defined as the number of correctly classified objects in a category divided by the total number of actual objects in that category. Precision is defined as the number of correctly classified objects in a category divided by the total number of objects that have been classified into that category.

**Experimental Implementation and Validation**

For testing and evaluation, the proposed method was empirically implemented in classifying IFC objects collected from 5 IFC models, with 5 additional objects from National BIM Library of UK (NBS of UK 2014). The implementation details are described in the following sections.

***(1) Data collection: collect IFC models from different sources to create a dataset with a broad coverage of different types of IFC entity usage.***

To cover the identified main types of AEC objects including beams, columns, footings, slabs, and walls in different types of representations and uses of IFC entities, the authors collected data from three different sources: (1) the "Common Building Information Model Files" published by buildingSMARTalliance of the National Institute of Building Sciences (East 2013), (2) Revit models exported as IFC data files, and (3) National BIM library of UK (NBS of UK 2014). Among the collected data, the authors selected the duplex apartment model (*Deplex_A*) from the first source, the Revit architectural sample model (*Rac_basic*), the Revit advanced structural sample model (*Rst_advanced*), the Revit basic structural sample model (*Rst_basic*), and the Revit technical school sample structural model (*Tech_school*) from the second source, and five special beam model objects (including four U-beams and one L-beam) from the third source. The selection was based on the variation in their model types and object types. The similarity between all the selected models (except for the special beam model objects) was that they all contained beams, columns, footings, slabs, and walls. The authors collected the objects by searching through open source BIM data and observing their included AEC objects. The selected models contained hundreds of AEC objects on average. The special beam model objects were collected so that the authors could test secondary development in addition to primary development of the proposed method.

***(2) Preprocessing: extract IFC objects from the collected models, manually label the data, and divide the objects into training set and testing set.***

All objects from the selected IFC models were extracted, resulting in a total of 1,891 objects. As shown in Table 1, there were 86, 85, 883, 446, 386, and 5 objects from each of the five IFC models and the National BIM library of UK, respectively.

The authors invited 3 independent annotators to manually label the same set of objects with their building element types. The average inter-annotator agreement was 87.21% initially (Table 2). For the objects that had different labels by different annotators, the authors arranged discussions with the annotators and tried to get agreement through debating and convincing each other. In the end, an average inter-annotator agreement of 99.06% was achieved. For the 18 objects (0.94% of the data) that annotators still did not achieve agreement, the authors picked the majority labels (examples in Table 3). BIM viewer was used to visualize the extracted objects and display their properties during the manual labeling. Based on the above labeling process, 795 objects were labeled as beams, 412 objects were labeled as columns, 348 objects were labeled as footings, 74 objects were labeled as slabs, and 262 objects were labeled as walls (Table 4). Using the data dividing Java program that the authors wrote, the authors collected 1,325 objects into the training set and 566 objects into the testing set. The collected data was not exhaustive but sufficient for testing the authors' proposed method (Beleites et al. 2013). In addition, the method can be used to continuously develop more patterns and rules to cover more categories when fed with more data. Because of the composite nature of the proposed method, the patterns and rules to be developed for future categories will not affect the processing results of the already covered categories.

***(3) Environment setup: initially build a classification algorithm with no rules or patterns.***

The authors developed the framework of the classification algorithm and implemented it in Java programming language. It takes a file as input and outputs a string that represents the classification result of the object in that file. If the object cannot be classified, an error message with detailed information will be displayed. At this step, there were no rules or patterns yet, and the default error message "cannot be classified" would be displayed if the method was applied to a model.

***(4) Primary development: study the representations of the training set objects in IFC, build rules and develop sub-algorithms to classify objects into existing categories in IFC.***

Using the iterative process described in the method section, the authors developed sub-algorithms and rules to classify all objects in the training data into five main existing IFC categories: beams, columns, footings, slabs, and walls. Among the instances in the dataset, the study of one object will usually be sufficient to classify all object instances with similar geometric representations. To study the effect of training at each stage of the development, the authors recorded the number of correctly classified instances after each stage of development and plotted them as a learning curve. Table 5 shows the geometric content of the study, and the number of correctly classified instances with respect to each stage of the development. In each stage, the geometric features of the targeted type of geometric representation were analyzed and used to compose patterns and rules for identifying objects represented using this targeted type of geometric representation. Stage 1 to Stage 8 focused on the "Swept Solid" type of geometric representation. Specifically, Stage 1 focused on rectangular shapes; Stage 2 focused on I-beams represented by "Swept Solid" with *IfcArbitraryClosedProfileDef*; Stage 3 focused on slabs represented by "Swept Solid" with *IfcArbitraryClosedProfileDef*; Stage 4 focused on objects represented by "Swept Solid" with *IfcCircleProfileDef*; Stage 5 focused on objects represented by "Swept Solid" with four other built-

in shape profiles, including I-shape, C-shape, U-shape, and L-shape; Stage 6 focused on objects represented by "Swept Solid" with built-in *IfcCircleHollowProfileDef*, which defines a ring shape; Stage 7 focused on objects represented by "Swept Solid" with *IfcCompositeCurve*; Stage 8 focused on objects represented by "Swept Solid" with *IfcClosedShell*. Stage 9 to 12 focused on "Brep," "Clipping," "CSG," and "Mapped Representation" types of geometric representation, respectively. Fig. 8 shows the plot of the learning curve. Some development details are described below.

In the geometric representations of objects in the training dataset, the following solid representation methods were used: "Swept Solid" (using *IfcExtrudedAreaSolid*), "Clipping," "MappedRepresentation," "Brep," and "CSG". Among these representation methods, "Swept Solid" is the most frequently used one: 1,035 of 1,325 (78.11%) of entities in the training data used "Swept Solid". This representation method extends a 2D shape through a direction that is not in the 2D plane, to create a 3D shape. For example, extending a long narrow rectangular shape on the floor vertically upwards creates a solid cuboid shape that could be used to represent the geometry of a vertical standing wall. The same wall may also be represented using a "Brep" by enclosing six connected faces. "Brep" is a powerful geometric representation in IFC (Alain 2016). It can be used to approximate almost any shape. The internal structure of Brep data can vary a lot, which adds to the complexity of Brep-based geometries and their processing. In contrast, "Swept Solid" (or *IfcExtrudedAreaSolid*) is a faster way to represent common building element shapes (buildingSMART 2018b). It can easily represent a cuboid shape by extending a rectangular planar surface in its normal direction or the opposite direction. There are also "Clipping", "CSG", and "MappedRepresentation" that can represent solid model elements (buildingSMART 2007). "Clipping" is the Boolean results of two representations; "MappedRepresentation" reuses existing representation for new ones; "CSG" is the Boolean results of multiple primitive solids.

In the first stage, the authors studied the data representation of "Swept Solid," which contains an instance of *IfcExtrudedAreaSolid* (buildingSMART 2018a). There are four attributes of an *IfcExtrudedAreaSolid*: a swept area, a direction, a position, and a depth. The swept area defines a 2D shape to be extended; the position defines the placement position and direction where the solid object is to be placed; the extruded direction defines a direction along which the swept area is extended; and the depth defines a distance for which the swept area is extended. The authors use the assumption that a beam, when represented using a "Swept Solid," is extended horizontally while other building elements such as walls and columns will be extended vertically. So the extruded direction is used as an indicator for differentiating beams from the other building elements. When looking at the orientation of an object, it is possible that an object is represented by "Swept Solid" with extrusion in the vertical direction but then rotated horizontally during the placement of the object. In other words, the position and the direction that an object was placed also need to be taken into consideration. As a result, in developing the algorithm, the authors combined both information.

In a "Swept Solid" representation, the extruded direction can be obtained from the *IfcDirection* property, and the placement is defined using an *IfcAxis2Placement3D,* as described by a point and two axes (ideally orthogonal). The point is the origin and the two axes are the Z and X axes. The axis Z = ($Z_0$, $Z_1$, $Z_2$) and X = ($X_0$, $X_1$, $X_2$) are both represented by a vector with three parameters. They are called "Axis" and "RefDirection" respectively in an *IfcAxis2Placement3D*. In this way, it defines a unique position and orientation for the placement of an object. In comparison, the direction of the original extruded direction in the "Swept Solid" representation is defined directly using a 3D vector (x, y, z) with three parameters. After extracting these information, the authors

combined the extruded direction and placement information using Equation (1) to compute the final extruded direction of the object.

Final extruded direction:

$$(x_{new}, y_{new}, z_{new}) = x^*(x0,x1,x2) + y^*(x0,x1,x2) \times (z0,x1,x2) + z^*(z0,z1,z2) \quad (1)$$

where:

Extruded direction $(x,y,z) = x * (1,0,0) + y * (0,1,0) + z * (0,0,1)$,

Placement Z axis (Axis): $(z0,z1,z2)$,

Placement X axis (RefDirection): $(x0,x1,x2)$,

As a result, if the final extruded direction is horizontal, the object will be processed as a candidate of a beam. In contrast, an object with vertical extruded direction will become a candidate for the other categories: column, footing, slab, and wall. Then the depth information could be used to differentiate the slab category from the other three categories. Finally, the 2D shape of the swept area (i.e., cross section) and ratios between different dimensions are used to differentiate the column, footing, and wall categories.

Fig. 9 shows the algorithm after development. The algorithm starts from a single IFC object and extracts its geometric representation by tracing its associated *IfcShapeRepresentation* instance. According to the extracted geometric representation type, the algorithm flows to "Clipping", "Swept Solid", "Brep", "MappedRepresentation", or "CSG". For example, if the geometric representation is a swept solid, the algorithm will extract its extruded direction. If the extruded direction is horizontal, the algorithm will check the geometry and classify the input into designated beam types; if the extruded direction is vertical, the algorithm will make the object a candidate of column, footing, slab, and wall categories. Based on the value of the extruded depth, slab can be

differentiated from the other three categories. To distinguish footing, column, and wall, the shape of the cross section (e.g., square v.s. circle) and ratios between the three dimensions are used. For example, a circular cross section profile excludes the object from the wall category. The dimensional ratio between height and the other two dimensions can be used to distinguish slabs from other categories.

For "Brep", the algorithm extracts the number of faces first. Using the number of faces, the algorithm selects possible candidates. For example, an instance with 6 faces will be a candidate of a cuboid. Then the sub-algorithm for each shape will verify the features of that shape. An example of development will be shown in step (5).

For "Clipping", the algorithm picks the main element that will be cut or added. The classification result will follow the result of that element. This method works well because most clipping results are some modification of an existing "Swept Solid," which has already been classified based on its geometric representation.

For "MappedRepresentation", the algorithm will track the original element to be mapped and classify it. The classification result of the original element will be used as the classification of the mapped object. Such use is feasible because the mapping from the original element to the mapped object does not change the internal geometric representation of the shape.

As a result of the development, the authors created an algorithm to classify an IFC object into one of the following building elements: beams, columns, footings, slabs, and walls. The algorithm did not use the entity name of the IFC object, because the entity name may be incorrect due to a misuse. Instead, it directly searches for the geometric representation information of the object from the standalone IFC file and uses this information for the classification.

***(5) Secondary development: study the representations of IFC objects and develop sub-algorithms to classify them into non-IFC defined categories.***

In this step, the authors expanded the classification on beam into subtypes of beams. Beams can be classified by its support into simply supported beam, fixed beam, cantilever beam, continuously supported beam, or by the cross-sectional shape into I-Beam, C-Beam, T-Beam, etc. (Chennu 2017; buildingSMART 2018c). Because the authors focus on using geometric information, which may not necessarily have the support type information, sub-algorithms were developed to classify the beams by their cross-sectional shapes.

In the collected data, there were three ways to represent the geometry of a beam: a "Swept Solid" with built-in 2D shapes, a "Swept Solid" with a 2D *IfcClosedShell*, and a "Brep," i.e., an *IfcFacetedBoundaryRepresentation*. The authors developed sub-algorithms for processing all the three cases.

In the first case, the beam's geometry will be represented by a "Swept Solid" with built-in 2D shape profiles, which is an *IfcProfileDef* (buildingSMART 2018c), such as *IfcIShapeProfileDef* and *IfcCShapeProfileDef*. Using shape profiles provided in IFC, it is straightforward to represent common shaped beams. For example, I-Beam can be represented using the *IfcIShapeProfileDef*, which can then be automatically classified as an I-Beam based on its profile shape name.

However, built-in shape profile types are not the only way to represent an I-Shape. In practice, there are a large amount of data that were using *IfcArbitraryClosedProfileDef*, which is a 2D shape bounded by some arbitrary lines or curves that are closed. For the closed curves, their 2D features can be used to identify the unique cross-sectional shape. For example, an I-Beam has two possible cross sections: W-Section and S-Section as shown in Fig. 10.

Although an S-Section can be classified as an I-Beam, the W-Section is much more common in industrial use. In fact, in the collected data, there were only W-Section I-Beams. Similar to I-Beam having different variants, there can be variants of W-Section I-Beams. However, the goal here was only to distinguish I-Beam from other beam types, such as C-Beams and U-Beams. To distinguish them, the authors developed a sub-algorithm that counts the number of boundary lines and curves and checks the linkages between them. The authors call this type of sub-algorithm shape recognizer. For example, a typical I-Beam cross section contains 12 lines (i.e., straight lines) and 4 curves. The linkages between the lines and curves are unique, which makes them feasible for use in distinction. For example, for a standard U-Beam as shown in Fig. 11, there are 8 lines. For a standard C-Beam, there are 12 lines and 8 curves. However, in the practical use of IFC, a C-Beam may only contain 12 lines and 4 curves or 12 lines without any curves, according to the level of details of their representations. Such complexity may increase the number of possible configurations of beam shapes. However, even in these special cases, shape configurations can still be enumerated. To sort the beams into different types, the authors developed the following four-step method (Fig. 12): (1) Input reading: read in a beam candidate; (2) Lines and curves counting: count the number of lines and curves of the geometric representation of the beam candidate; (3) Shape checking: compare the line configuration of the geometric representation with all studied 2D shapes; (4) Linkage verification: verify the possible shapes by checking the unique linkages between lines and curves.

Among all the beams in the training data, the authors studied four shapes that had built-in 2D profiles in IFC. Examples of these shapes are shown in the beams in Fig. 11. They are *IfcIShapeProfile*, *IfcCShapeProfile*, *IfcUShapeProfile*, and *IfcLShapeProfile* (buildingSMART

2018c). They can be recognized by adding a new count of the lines and curves, and a verification of their linkages.

Table 6 shows the calculated possible counts of lines and curves of Rectangular Beam, U-Beam, C-Beam, I-Beam, and L-Beam, respectively.

Based on the information in Table 6, the authors summarized possible beam types of different geometric patterns in terms of the number of lines and curves in their geometric representations (Table 7).

According to Table 7, with the same number of lines and curves in their geometric representations, two beam objects may still have different possible beam types. To successfully differentiate such types of beams, the linkage types of the lines and curves in the geometric representations were used. For example, for an I-Beam, the following three aspects of the connections between lines will be checked. First, all the angles between connected lines must be right angles. Second, there must be four different lengths of lines based on the symmetry of I-Beam. Third, because lines have directions in IFC data, the way they are connected (e.g., left turn versus right turn) also provide useful information. By these observations, the authors developed a verification sub-algorithm that verifies the angles, lengths, and turn directions of lines and curves.

Conceptually, checking linkages helps differentiate the two shapes shown in Fig. 13. These two shapes both have 12 lines, with the lengths of all the lines being the same. Apparently, the shape in the right part of Fig. 13 should not be classified as an I-Beam while the shape in the left part of Fig. 13 should. Such distinction is made at the linkage checking step by analyzing the turn directions of the lines.

For Brep, there are many ways to represent a beam, the authors used a data-driven approach and developed several sub-algorithms for different shape representations. Each time a new shape representation was came across in the training data, a new sub-algorithm was added.

As a result, the authors developed sub-algorithms for all types of beams observed in the training data. There were three main types of sub-algorithms developed: one for "Swept Solid" with built-in shape profiles, one for "Swept Solid" with a 2D *IfcClosedShell*, and one for Brep.

The first sub-algorithm type was straightforward by tracking the built-in shape used, as previously discussed. The second sub-algorithm type was using the shape recognizer that differentiates shapes based on patterns of lines and curves. The third sub-algorithm type classifies the beam objects that are represented using "Brep." Some of them are single beams, for which a recognizer sub-algorithm was developed for each type of beam. Some of them are trusses. Fig. 14 shows a visualization of a truss. In the training data, there can be from 42 sub-elements to as many as 72 sub-elements in a truss. Each truss consists of two major beams (i.e., longeron) and many web members across the bridge, with each major beam consisted of two L-beams as shown in Fig. 14. The authors developed a sub-algorithm that: (1) recognizes the two major beams (therefore the four L-beams), and (2) verifies and counts the web members. The authors classified this type of "Beam" into a truss category. As shown in Fig. 15, this sub-algorithm takes a single IFC object as input and counts the number of sub-elements ($n$). If $n$ is between 42 and 72, then the sub-algorithm finds the two sub-elements with the largest two sizes (*se1* and *se2*) and checks their shapes. If $n$ is not between 42 and 72, then the object being processed is not identified as a truss. In the shape checking, the sub-algorithm tests if any shape associated with the two sub-elements is not in L-shape. If so, then the object being processed is not identified as a truss. Otherwise (all the four shapes associated with the two sub-elements are in L-shape), store all the remaining $n-2$ sub-

elements (i.e., web members) into a stack structure (*s*). The content in stack *s* is checked, if *s* is not empty, the sub-algorithm pops one sub-element from *s* and checks its position and number of faces. If the position is between the positions of *se1* and *se2*, and at the same time if the number of faces is among 6, 8, 10, and 14, then this sub-element passes the test and the sub-algorithm moves on to test the next sub-element from stack *s*. If all sub-elements from stack *s* pass the test, then the object being processed is identified as a truss.

Similar to truss, the authors developed sub-algorithms for each unique type of beams. Examples of such shapes of beams are shown in Fig. 16.

***(6) Error analysis and training improvement: analyze errors in the classification results on the training set, further add/revise sub-algorithms and rules to improve the training performance.***

The results of object classification were evaluated in terms of recall and precision (Table 8).

In the categories of beam, slab, and wall, 100% recall and precision were achieved. The authors noticed a low recall (31.28%) in the footing category and low precision (63.05%) in the column category. Through analysis, it was found that the shapes of 167 footings in the experiment were similar to columns and incorrectly classified into columns, as shown in Fig. 17. Therefore the cause of this error was the insufficiency of relying solely on geometric information in such a case. In other words, by checking shape information here, the footing cannot be differentiated from columns. A possible solution to that is adding in the consideration of other types of information such as relative location of an object with other objects. Other than the 167 misclassified footings, the classification results were 100% in both recall and precision.

The classification results on detailed beam types achieved 100% recall and precision in all categories, as shown in Table 9.

***(7) Testing: apply the developed classification algorithm to testing data for evaluation.***

To test the expected performance of the algorithm, the authors tested the algorithm on testing data. The results are listed in Table 10 and Table 11.

**Results Analysis and Discussion**

The developed algorithm worked well in most cases on the testing data with a higher than 90% precision and recall. A low recall (20.95%) in the footing category was due to the lack of distinction between the shapes of footings and columns (as in the training data), which indicated the insufficiency of relying solely on geometric information for object classification in certain cases. By adding a simple elevation information, however, the algorithm successfully distinguished such footings from columns. For the rest of errors in testing data, the authors inspected the instances and found that all the 5 error instances were due to new geometric representations in the testing data which were not covered in the training data. Among these 5 error instances, one was due to a new "SurfaceModel" geometric representation instance in the testing data, and 4 were due to new "Brep" geometric representation instances in the testing data.

The experiment shows that our proposed method could be used to develop an algorithm (with sub-algorithms) that successfully captures the core features of object geometries and use them to distinguish AEC objects. The core features are consisted of: number of sub-components, number of faces, cross section profile, extrusion direction, dimensional ratio, number of straight lines and curves, line connection angle, length, and turn direction. Using these features, the algorithm can correctly classify beams, columns, footing, slabs, and walls, where the errors come from either different objects sharing the same exact shape or the lack of coverage of geometric patterns in the training data. For beams, the algorithm can identify the detailed beam types it was designed to identify with 100% precision and recall. The algorithm can be further extended if more objects of

different types and shapes are added to the training data. It can be continuously and accumulatively developed in this manner by adding more patterns and rules to cover more categories to ultimately lead to a comprehensive classification algorithm that identifies any type of AEC object in IFC automatically. Because of the composite nature of the proposed method, the patterns and rules to be developed for future categories will not affect the processing results of the already covered categories, therefore, the authors' proposed method can result in an accurate and reliable classification method.

A comparison between the proposed method and the methods by Ma et al. (2017) and Sacks et al. (2017) was conducted (Table 12). While all methods work for BIM and could achieve 100% precision and recall, their computational complexities differ. The method by Ma et al. (2017) has a time complexity of O(kn), where k is the highest number of properties for a studied object, and n is the number of studied objects. The method by Sacks et al. (2017) has a time complexity of O(n) in theory, where n is the total number of objects to be sort. In practice, the complexity can be higher because an optimal subset of unique rules may not always be achieved. In contrast, the algorithm developed using the proposed method in this paper has constant time complexity O(1), because the algorithm solely analyzes the geometric properties of the instances without the need of comparing an object with all possible categories in an enumerative manner. In addition, the proposed method does not require the use of reference objects during the classification application stage, which was needed in the methods of Ma et al. (2017) and Sacks et al. (2017).

**Limitations and Future Work**

In spite of the promising experimental results and the ability to achieve 100% recall and precision in automated AEC object classification, the following limitation of the proposed method is acknowledged. The proposed method is labor intensive in the algorithm development phase,

especially when a comprehensive algorithm is pursued which is expected to cover all possible geometric shape representations of AEC objects. In future work, the authors plan to: (1) formally add the use of locational information of AEC objects into the proposed method to avoid the confusion of objects of the same geometry but different types; (2) further develop the algorithm with more data that belong to more categories and cover more geometric shape representations of AEC objects; and (3) investigate potential ways to automate some or all steps of the algorithm development phase using machine learning especially deep learning.

**Contributions to the Body of Knowledge**

This research is important from both intellectual and application perspectives. From an intellectual perspective, this research contributes to the body of knowledge in four main ways. First, we offer a new data-driven method for developing an algorithm and sub-algorithms that can automatically classify IFC-based BIM objects into predefined categories. The algorithms rely on the inherent geometric features of AEC objects rather than entity or attribute names and therefore prevent classification errors caused by misuse of entities. Geometric features are stable and reliable properties of AEC objects (not changing with regard to software implementation, modeling decisions, an/or language/culture contexts) and therefore object classification algorithms developed using the authors' proposed method can be more robust than those that depend on entity or attribute names. Other than the cases where different types of AEC objects have the exact same shape, the algorithm could classify objects with 100% recall and precision. Second, we offer a set of features to capture the core geometric representation of AEC objects, including: number of sub-components, number of faces, cross section profile, extrusion direction, dimensional ratio, number of straight lines and curves, line connection angle, length, and turn direction. Experiments show that this set of features successfully capture the characteristics of the geometric representations of

AEC objects for distinguishing different objects. Third, we show that geometric information can be used to differentiate AEC objects in BIM, except for the rare cases where different objects have the same geometry. The impact of applying this work in the AEC domain could be far-reaching. First, the use of inherent geometric features of AEC objects in classifying the objects opens a new door to BIM interoperability because such features of AEC objects do not change according to modeler, software provider, language, culture or other contexts. Second, the elimination of human-induced misuse errors in BIM enables better usability of BIM in downstream applications such as cost estimation, building code compliance checking, and structural analysis. A better usability of BIM can in turn promote the adoption of BIM in the AEC industry. Third, the proposed method can be used by our research community to develop AEC object classification algorithms in a continuous and accumulatively way, which will ultimately lead to a comprehensive set of AEC object classification algorithms that will help significantly reduce or eliminate classification errors of BIM objects caused by misuse of entities.

**Conclusions**

This paper presented a data-driven, iterative method for automated classification of AEC objects in an IFC-based BIM. The method can be used to develop an algorithm that reads in an AEC object and automatically classifies it into predefined categories. These categories include existing IFC categories that represent common building object types and non-IFC categories that can represent a more detailed level of classification of objects. The developed algorithm consists of multiple sub-algorithms with each sub-algorithm depicting a pattern matching rule based on patterns of selected features. To test the proposed methodology, an experiment was conducted where IFC models were collected from three different sources, from which 1,891 objects were extracted and manually labeled with five IFC categories (beams, columns, footings, slabs, and walls) and eleven

non-IFC categories (e.g., C-Beam, I-Beam, L-Beam, U-Beam). The data was divided into training data (1,325 objects) and testing data (566 objects). An algorithm was developed using the proposed method based on the training data and tested on the testing data. In common building elements categories, 84.45% recall and 85.20% precision were achieved. In detailed beam categories, 100% recall and precision were achieved. For common building element categories, the errors were found to come from two sources: (1) different objects (i.e., footings and columns) sharing the exact same geometry; and (2) occurrence of geometric shape representation patterns in testing data that were not included in training data. The first source of error can be eliminated by adding locational information of objects for consideration in addition to geometric information. The second source of error can be avoided by including all foreseeable geometric shape representations in the training data. By such a strategy, the proposed method can achieve 100% recall and precision in the classification of all categories of AEC objects. The computational complexity of the authors' method was also compared with the state-of-the-art methods. The authors' method has a constant computational complexity which is better than the linear (or higher) computational complexity of the state-of-the-art methods.

**Acknowledgements**

The authors would like to thank the National Science Foundation (NSF). This material is based on work supported by the NSF under Grant No. 1745374. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the NSF.

**References**

Alain, H. (2016). "DRAGON5 and DONJON5, the contribution of École Polytechnique de Montréal to the SALOME platform." *Annals of Nuclear Energy*, 87(1), 12-20.

Balali, V., Ashouri Rad, A., and Golparvar-Fard, M. (2015) "Detection, classification, and mapping of U.S. traffic signs using Google street view images for roadway inventory management." *Visualization in Eng.*, 3(15), 1-18.

Beleites, C., Neugebauer, U., Bocklitz, T., Krafft, C., and Popp, J. (2013). "Sample size planning for classification models" *Analytica Chimica Acta*, 760, 25-33.Bo, L., Ren, X., and Fox, D. (2013). "Unsupervised feature learning for RGB-D based object recognition." *Experimental Robotics. Springer Tracts in Advanced Robotics*, 88, Springer, Heidelberg.

buildingSMART. (2007). "IFC2x edition 3 technical corrigendum 1." <http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/index.htm> (Jul 18, 2018).

buildingSMART. (2018a). "IFC Overview Summary." <http://www.buildingsmart-tech.org/specifications/ifc-overview> (Jul 18, 2018).

buildingSMART. (2018b). "IfcExtrudedAreaSolid." <http://www.buildingsmarttech.org/ifc/IFC2x3/TC1/html/ifcgeometricmodelresource/lexical/ifcextrudedareasolid.htm> (Jul 18, 2018).

buildingSMART. (2018c). "IfcProfileDef." <http://www.buildingsmart-tech.org/ifc/IFC2x3/TC1/html/ifcprofileresource/lexical/ifcprofiledef.htm> (Jul 18, 2018).

Chen, J., Fang, Y., and Cho, Y. (2016). "Automated equipment recognition and classification from scattered point clouds for construction management." *Proc., Inter. Symp. on Autom. and Rob. in Const.*, 218-255, Jul. 18, 2016, Auburn, AL.

Chennu, V. (2017). "Different types of beams." *ME mechanical.*

East, E.W. (2013). "Common building information model files and tools." < https://www.nibs.org/page/bsa_commonbimfiles?> (Jul 18, 2018).

Eastman, C., Teicholz, P., Sacks, R. and Liston, K. (2011). "BIM handbook: A guide to building information modeling for owners, managers, designers, engineers and contractors." John Wiley &Sons, Inc., 2nd edition, New Jersey.

Fang, W., Ding L., Luo H., and Love, P.E.D. (2018). "Falls from heights: A computer vision-based approach for safety harness detection." *Autom. in Const.*, 91, 53-61.

Fang, Y., Chen, J., Cho, Y., and Zhang, P. (2016). "A point cloud-vision hybrid approach for 3D location tracking of mobile construction assets." *Inter. Symp. on Autom. and Rob. in Const.*, Jul. 18-21, 2016, Auburn, AL.

Feng, C., Liu, M., Kao, C., and Lee, T. (2017). "Deep active learning for civil infrastructure defect detection and classification." *ASCE Inter. Workshop on Comput. in Civ. Eng.* 2017, ASCE, Reston, VA.

Geiger, A., Benner J., and Haefele, K.H. (2014). "Generalization of 3D IFC building models." *3D Geoinformation Science,* 19-35.

Golparvar-Fard, M., Balali, V., and de la Garza, J.M. (2013). "Segmentation and recognition of highway assets using image-based 3D point clouds and semantic texton forests." *J. of Comput. Civ. Eng.*, 29(1).

GopalakrishnanHoda, K., Gholami, H., Vidyadharan, A., Choudhary, A., and Agrawal, A. (2017). "Crack damage detection in unmanned aerial vehicle images of civil infrastructure using pre-trained deep learning model." *Inter. J. for Traffic and Transport Eng.*, 8(1), 1-14

Hamil, S. (2012). "The End Of Babel - IFC promotional video." <http://constructioncode.blogspot.com/2012/07/end-of-babel-ifc-promotional-video.html> (Jul 18, 2018).

Han, K., and Golparvar-Fard, M. (2015). "Appearance-based material classification for monitoring of operation-level construction progress using 4D BIM and site photologs." *J. of Autom. in Const.*, 53, 44-57.

Hefele, J. and Dolin, R.M. (1998). "Density effect of inspection data points in as-built modeling of parts." Los Alamos National Lab., New Mexico.

Henri, C. and Claire, L. (2017). "Handbook of categorization in cognitive science." Elsevier Ltd., 2st edition, Netherlands.

Huber, D., Akinci, B., Adan, A., Anil, E., Okorn B., and Xiong X. (2011). "Methods for automatically modeling and representing as-built building information models." *Proc., NSF CMMI Res. Innovation Conf*, Jan. 4-7, 2011, Atlanta, GA.

Kalasapudi V.S., Tang, P., Zhang, C., Diosdado, J., and Ganapathy, R. (2015). "Adaptive 3D Imaging and Tolerance Analysis of Prefabricated Components for Accelerated Construction." *Procedia Eng.,* 118, 1060–1067.

Kemal, P. and Salih, G. (2007). "Detection of ECG arrhythmia using a differential expert system approach based on principal component analysis and least square support vector machine." *Applied Mathematics and Computation*, 186(1), 898-906.

Koo, B., and Shin, B. (2018) "Applying novelty detection to identify model element to IFC class misclassifications on architectural and infrastructure Building Information Models." *J. Comput. Design and Eng.*, 5(4), 391-400.

Liu, K., and Golparvar-Fard, M. (2015). "Crowdsourcing construction activity analysis from jobsite video streams." *J. of Const, Eng. and Manag.,* 141(11).

Ma, H., Elsa, K., Chung, C., and Amor, R. (2006). "Testing semantic interoperability." *Proc., Joint Int. Conf on Comput. and Decision Making in Civ. and Build. Eng.,* Jun. 14-16, 2006, Montreal, Canada.

Ma, L., Sacks, R., Kattel, U., and Bloch, T. (2017). "3D object classification using geometric features and pairwise relationships." *Computer-Aided Civil and Infrastructure Eng.,* 33(2018), 152-164.

Makhoul, J., Kubala, F., Schwartz, R., and Weischedel, R. (1999). "Performance measures for information extraction." *Proc., DARPA Broadcast News Workshop,* Morgan Kaufmann, San Francisco, California, 249-252.

Memarzadeh, M., Golparvar-Fard, M., and Niebles, J.C. (2013). "Automated 2D detection of construction equipment and workers from site video streams using histograms of oriented gradients and colors." *J. of Autom. in Const.,* 32, 24-37.

Narazaki, Y., Hoskere, V., Hoang, T.A., Spencer, B.F.Jr. (2018). "Automated vision-based bridge component extraction using multiscale convolutional neural networks." arXiv:1805.06042 [cs.CV].

NBS of UK. (2014). "National BIM Library." <http://www.nationalbimlibrary.com/> (Jul 18, 2018).

Pazlar, T. and Turk, Z. (2008). "Interoperability in practice: geometric data exchance using the IFC standard." *J. of Inform. Techno. in Const.,*13, 362-380.

Qin, F., Li, L., Gao, S., Yang, X., and Chen, X. (2014). "A deep learning approach to the classification of 3D CAD models." *J. Zhejiang University SCIENCE C,* 15(2), 91-106.

Quirk, V. (2012). "A brief history of BIM." <https://www.archdaily.com/302490/a-brief-history-of-bim> (Jul 18, 2018).

Richard, S., Brody, H., Bharath, B., Christopher, D.M., Andrew Y. N. (2012). "Convolutional-recursive deep learning for 3D object classification." *Proc., 25th Inter. Conf. on Neural Inform,* Springer, New York, NY, 656-664.

Sacks, R., Ma, L., Yosef, R., and Borrmann, A. (2017). "Semantic enrichment for building information modeling: procedure for compiling inference rules and operators for complex geometry." *J. of Comput. Civ. Eng.*, 31(6).

Tang, P., and Akinci, B. (2012). "Formalization of workflows for extracting bridge surveying goals from laser-scanned data." *Autom. in Const.,* 22(3), 306–31.

Ullman, S (2007). "Object recognition and segmentation by a fragment-based hierarchy." *Trends in Cognitive Sciences,* 11 (2), 58-64.

Ullman, S. and Epshtein, B. (2007). "Visual classification by a hierarchy of extended fragments." *Toward Category-Level Object Recognition,* 321-344.

Voegtle, T. and Steinle, E. (2003). "On the quality of object classification and automated building modeling based on laserscanning data." *IAPRSIS,* 34(13), 49-155.

Wang, C. and Cho, Y. (2014). "Automatic as-is BIM creation from unorganized point clouds." 2014 *ASCE Constr. Res. Congress (CRC),* ASCE, Reston, VA, 917-924.

Wang, G., Hoiem D., and Forsyth, D.A. (2009). "Building text features for object image classification." *2009 IEEE Conference on Computer Vision and Pattern Recognition,* IEEE, New York, NY, 1367-1374.

Wang, Z. and Schenk, T. (2010). "Building extraction and reconstruction from Lidar data." *Inter. Archives of Photogrammetry and Remote Sensing,* 33(3), 958-964.

Won, J., Lee, G., and Cho, C. (2013). "No-schema algorithm for extracting a partial model from an IFC instance model." *J. of Comput. Civ. Eng.*, 27(6), 585-592.

Zhang, J. (2018). "Towards systematic understanding of geometric representations in BIM standard: an empirical data-driven approach." *Proc., 2018 Constr. Res. Congress (CRC)*, ASCE, Reston, VA, 96-105.

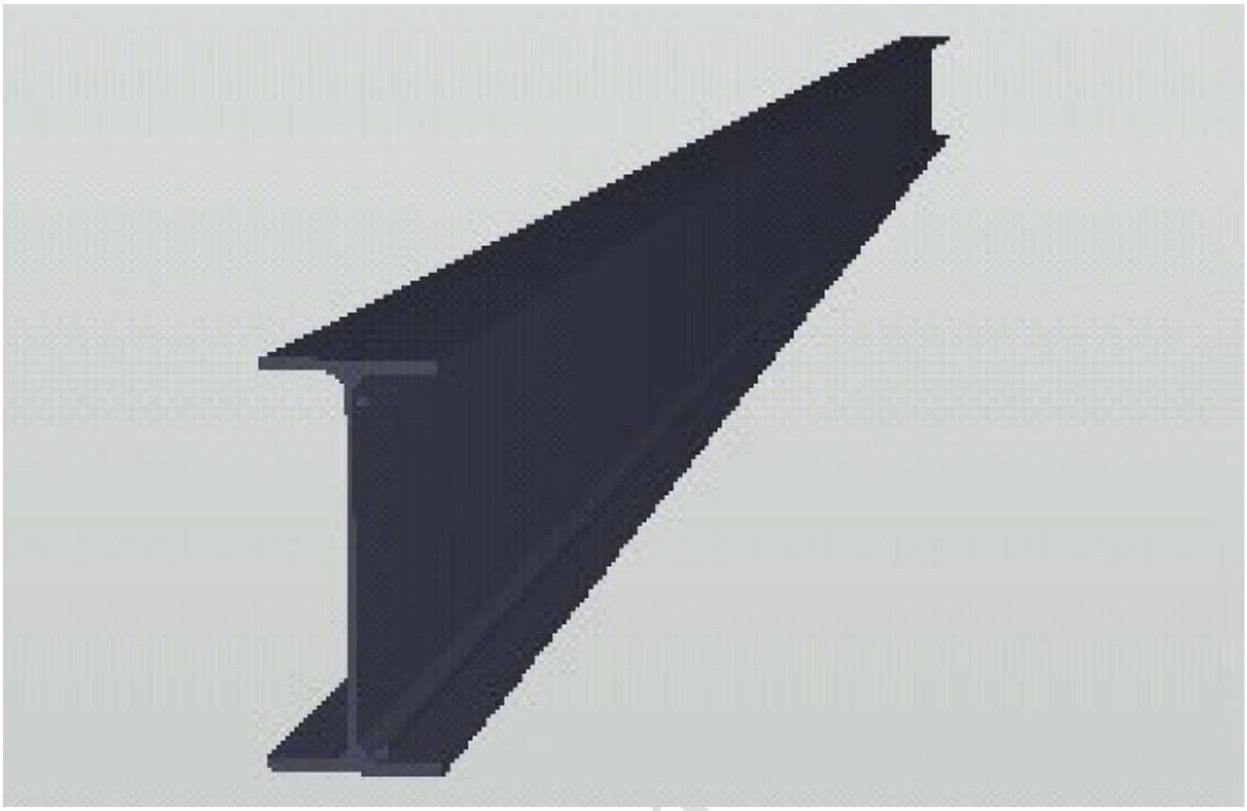

**Fig. 1.** The visualization of a wide flange beam/I-beam

```
DATA;
#1= IFCBEAM('2OrWItJ6zAwBNp0OUxK$8W',#2,'M_W-Wide Flange:W410X60:W410X60:208949',$,'M_W-
Wide Flange:W410X60:208814',#3,#4,'208949');
#2= IFCOWNERHISTORY(#5,#6,$,.NOCHANGE.,$,$,$,0);
#3= IFCLOCALPLACEMENT(#7,#8);
#4= IFCPRODUCTDEFINITIONSHAPE($,$,(#9,#10));
#5= IFCPERSONANDORGANIZATION(#11,#12,$);
#6= IFCAPPLICATION(#13,'2011','Autodesk Revit Architecture 2011','Revit');
#7= IFCLOCALPLACEMENT(#14,#15);
#8= IFCAXIS2PLACEMENT3D(#16,#17,#18);
#9= IFCSHAPEREPRESENTATION(#19,'Axis','Curve2D',(#20));
#10= IFCSHAPEREPRESENTATION(#19,'Body','SweptSolid',(#21));
```

**Fig. 2.** IFC data of an I-beam as represented by an *IfcBeam*

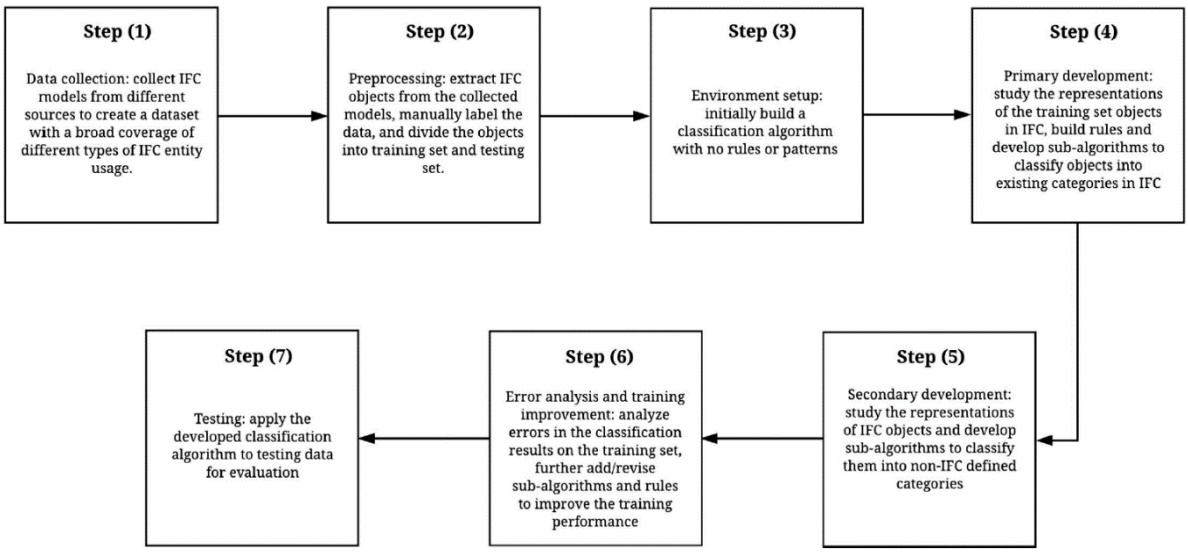


**Fig. 3.** The proposed 7-step method for automated IFC-based BIM object classification

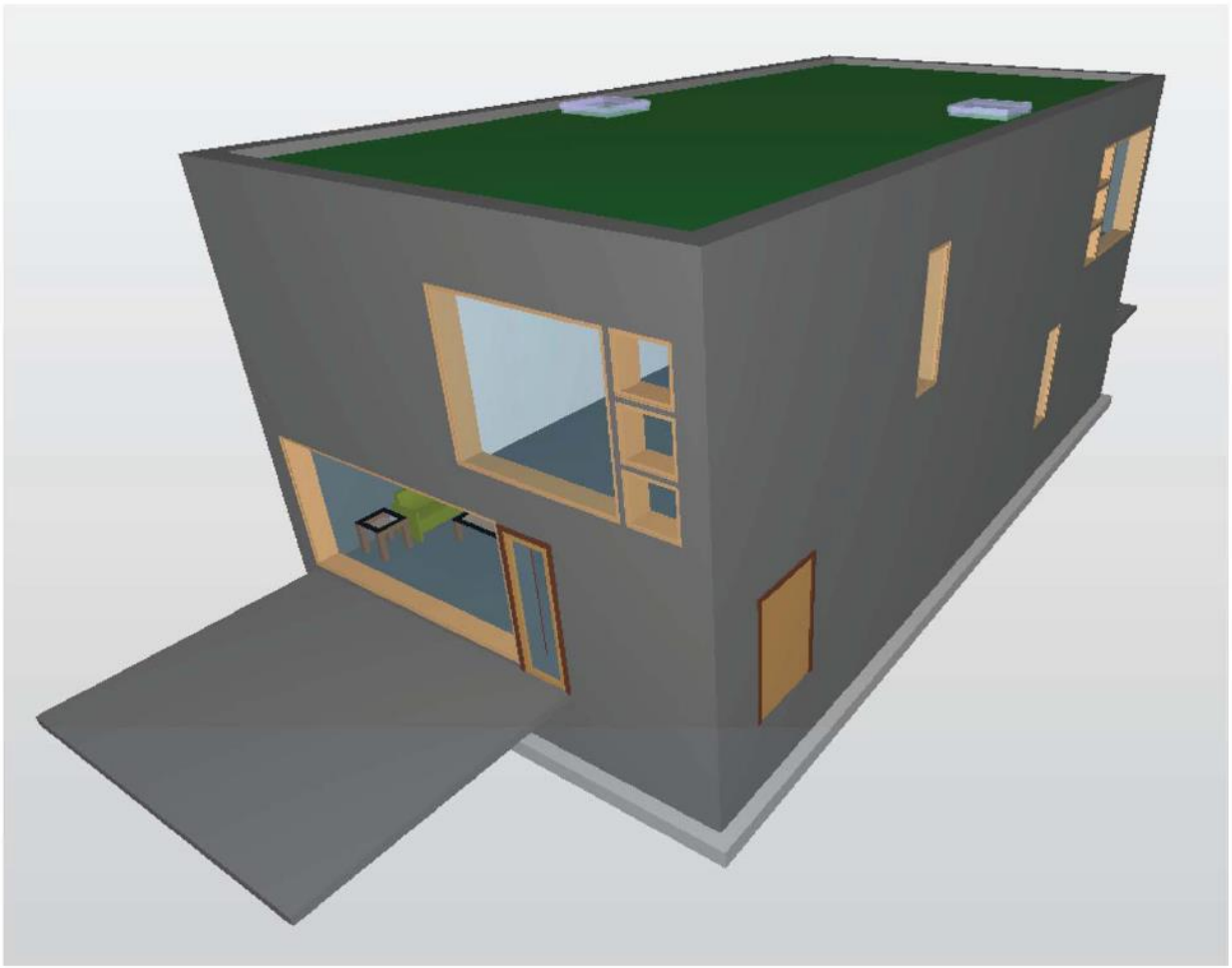

**Fig. 4.** Visualization of a Duplex Apartment model

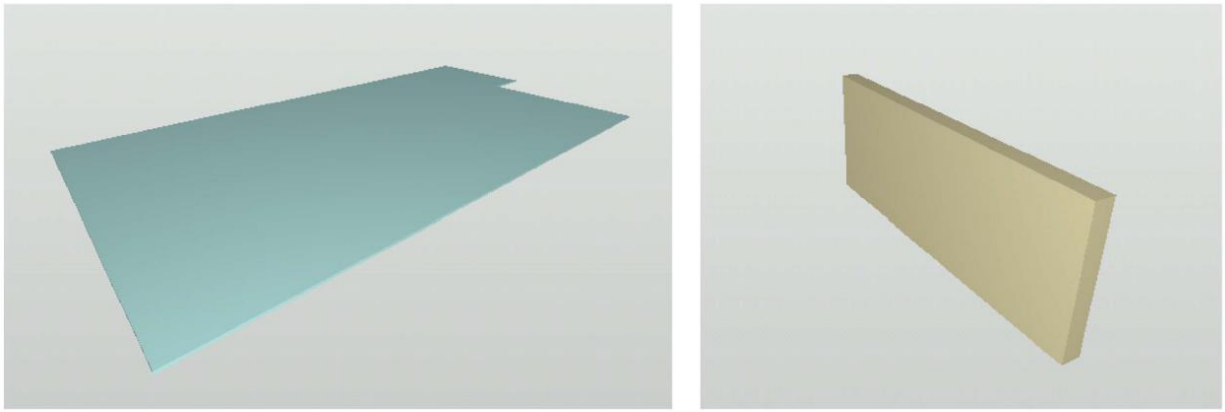


**Fig. 5.** Extracted building elements from the Duplex Apartment model: a slab (left) and a wall (right)

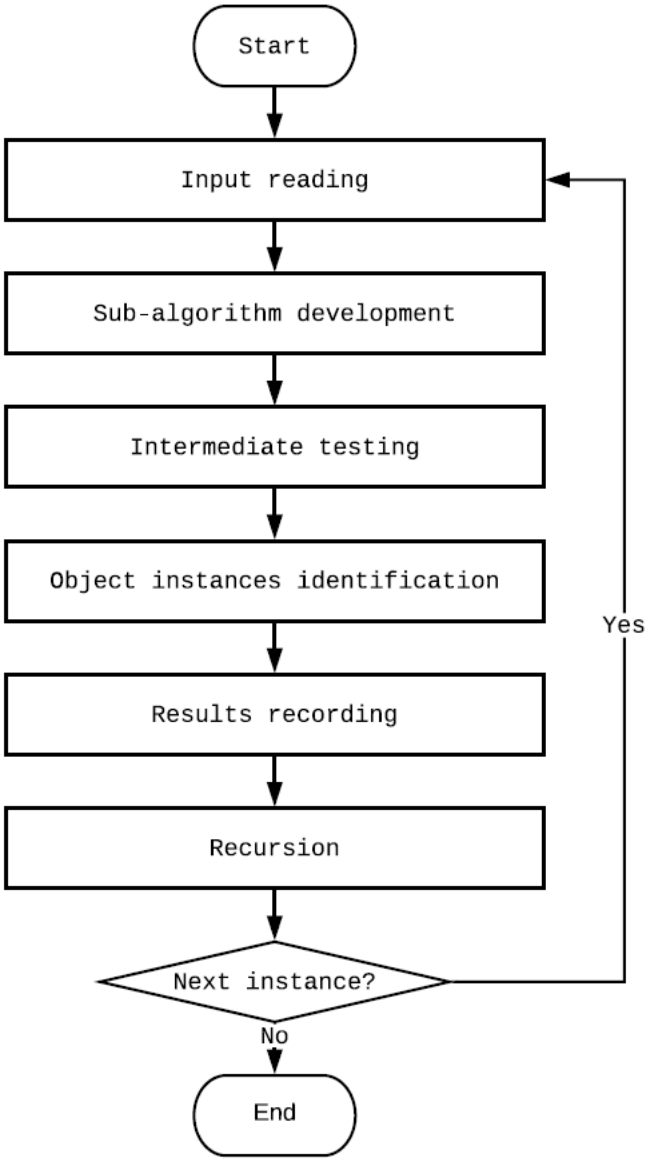


**Fig. 6.** Primary development flowchart

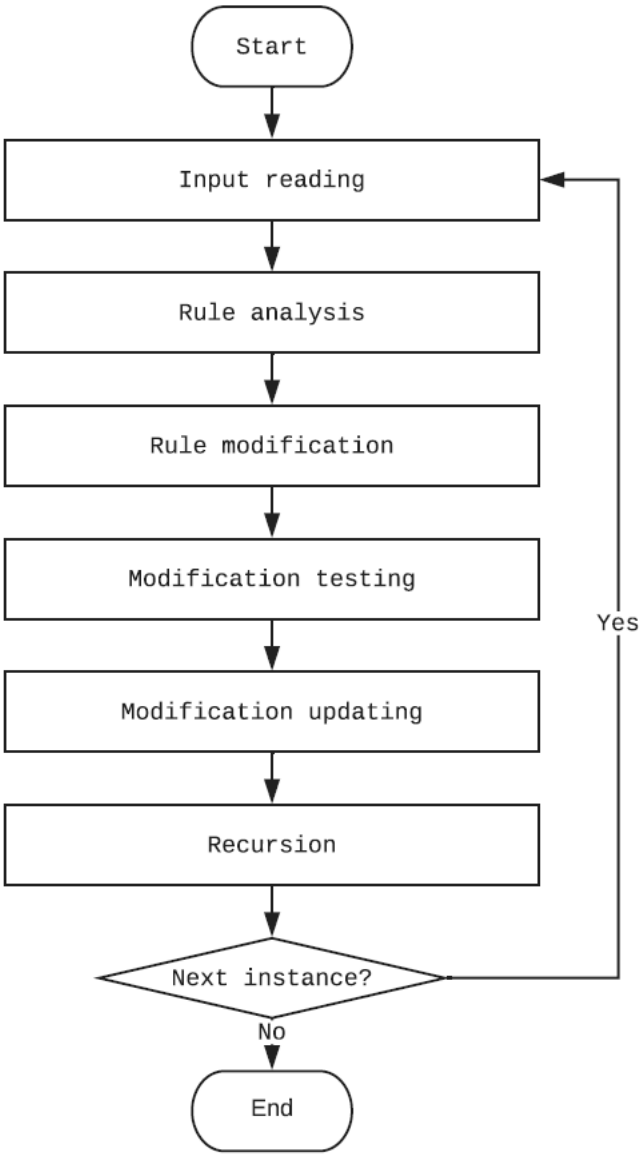


**Fig. 7.** Error analysis and training improvement flowchart

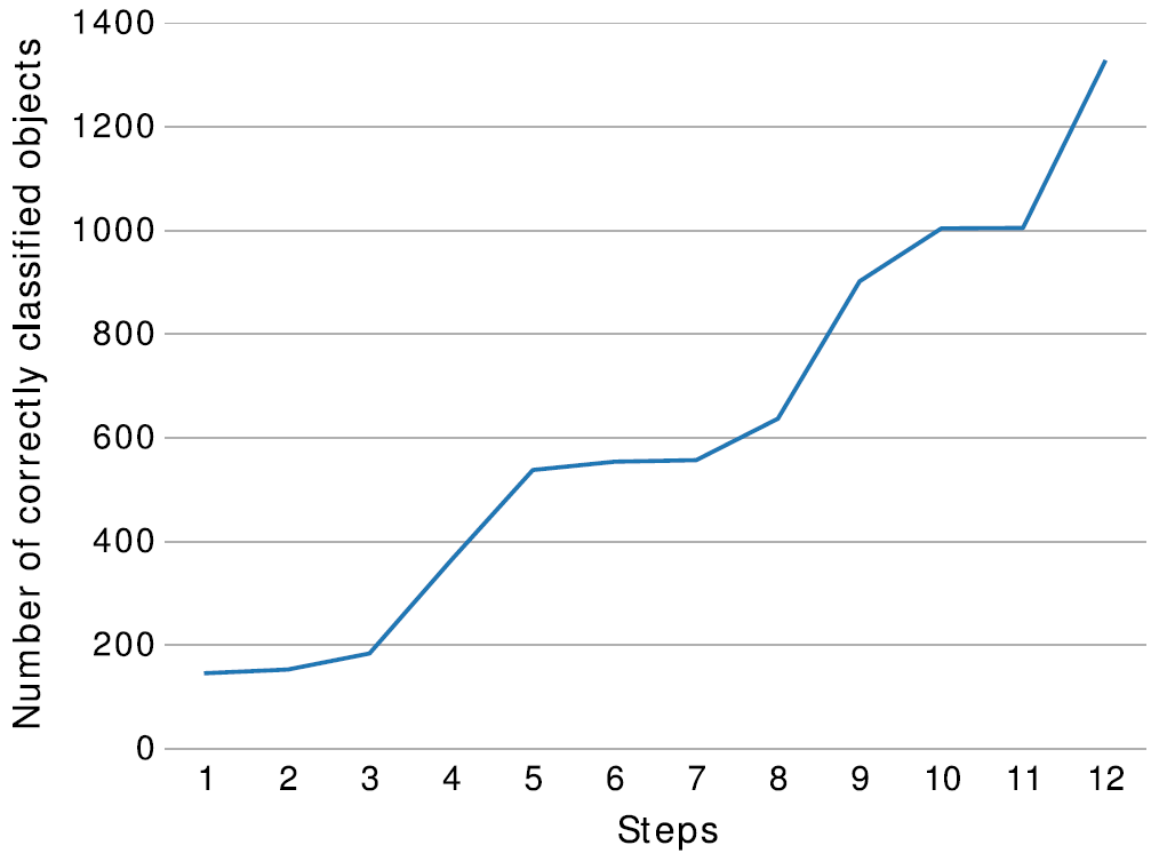


**Fig. 8.** Plot of the learning curve for all objects in the training data

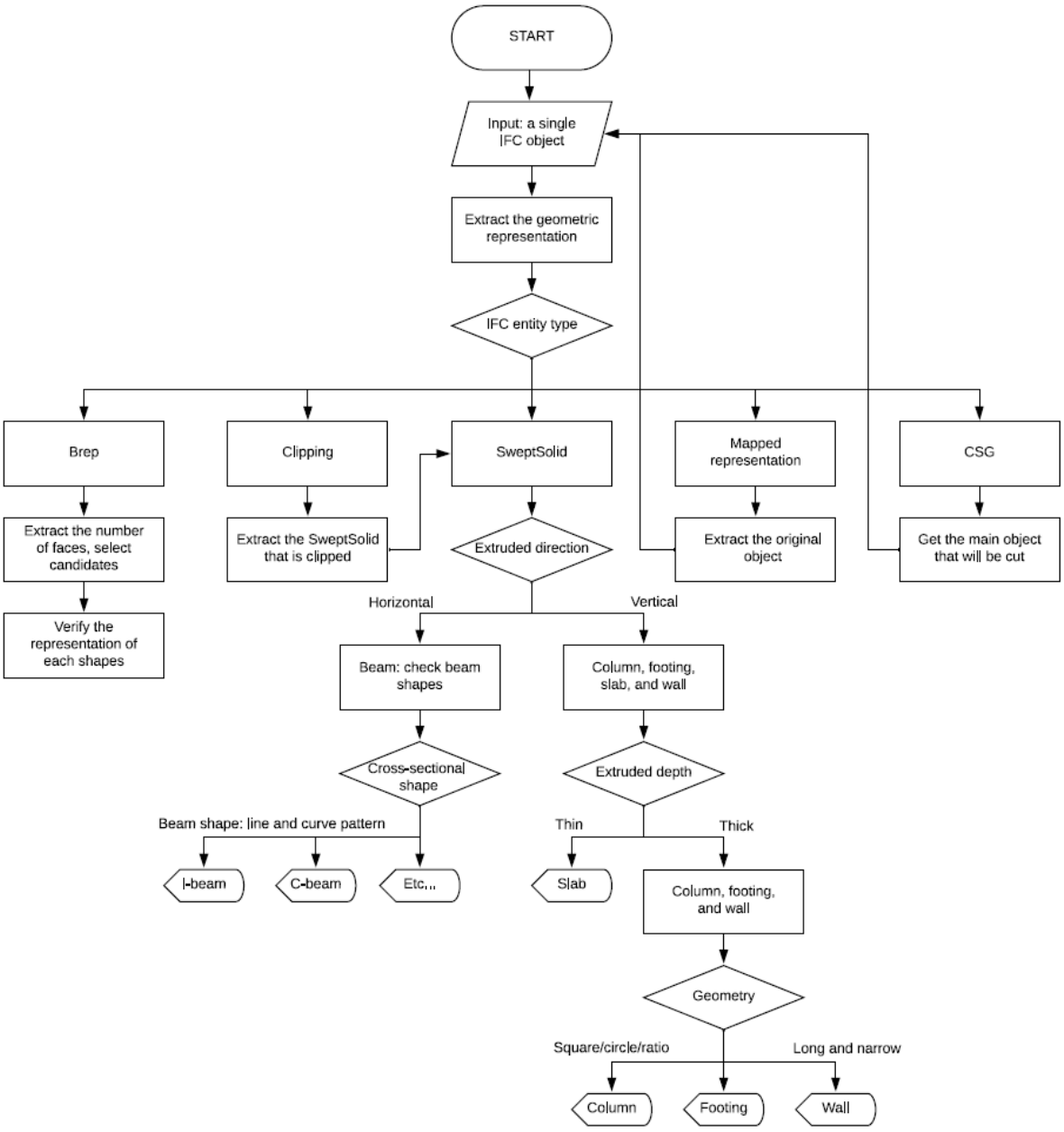


**Fig. 9.** Algorithm flowchart for object classification

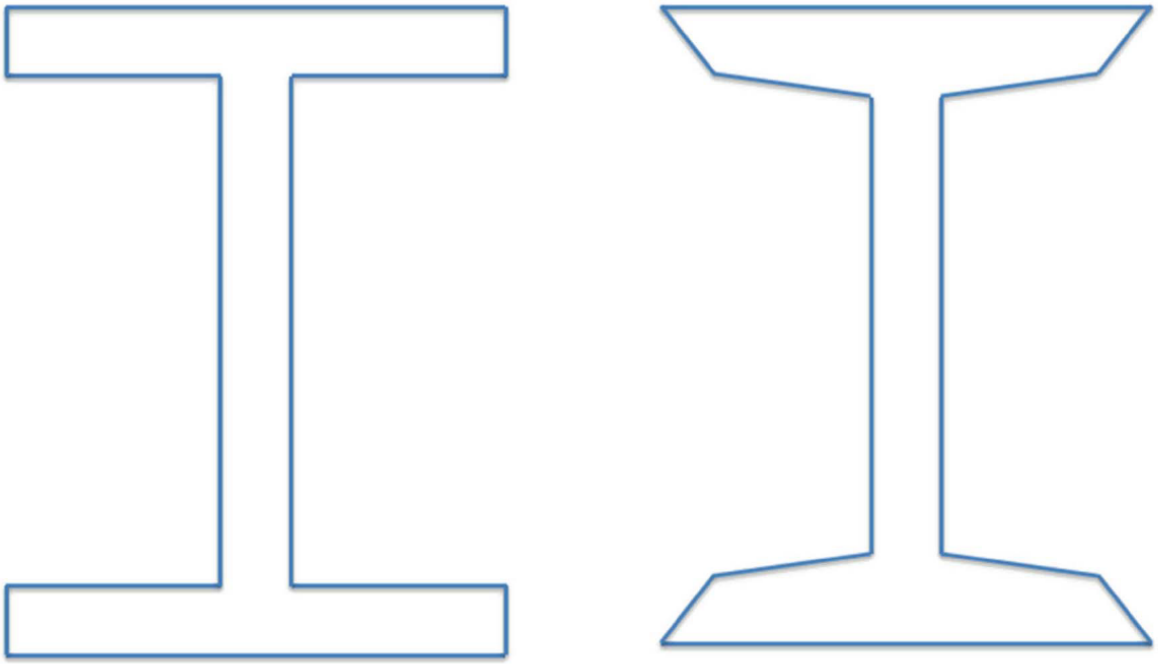

**Fig. 10.** W-Section (left) and S-Section (right) types of I-beam

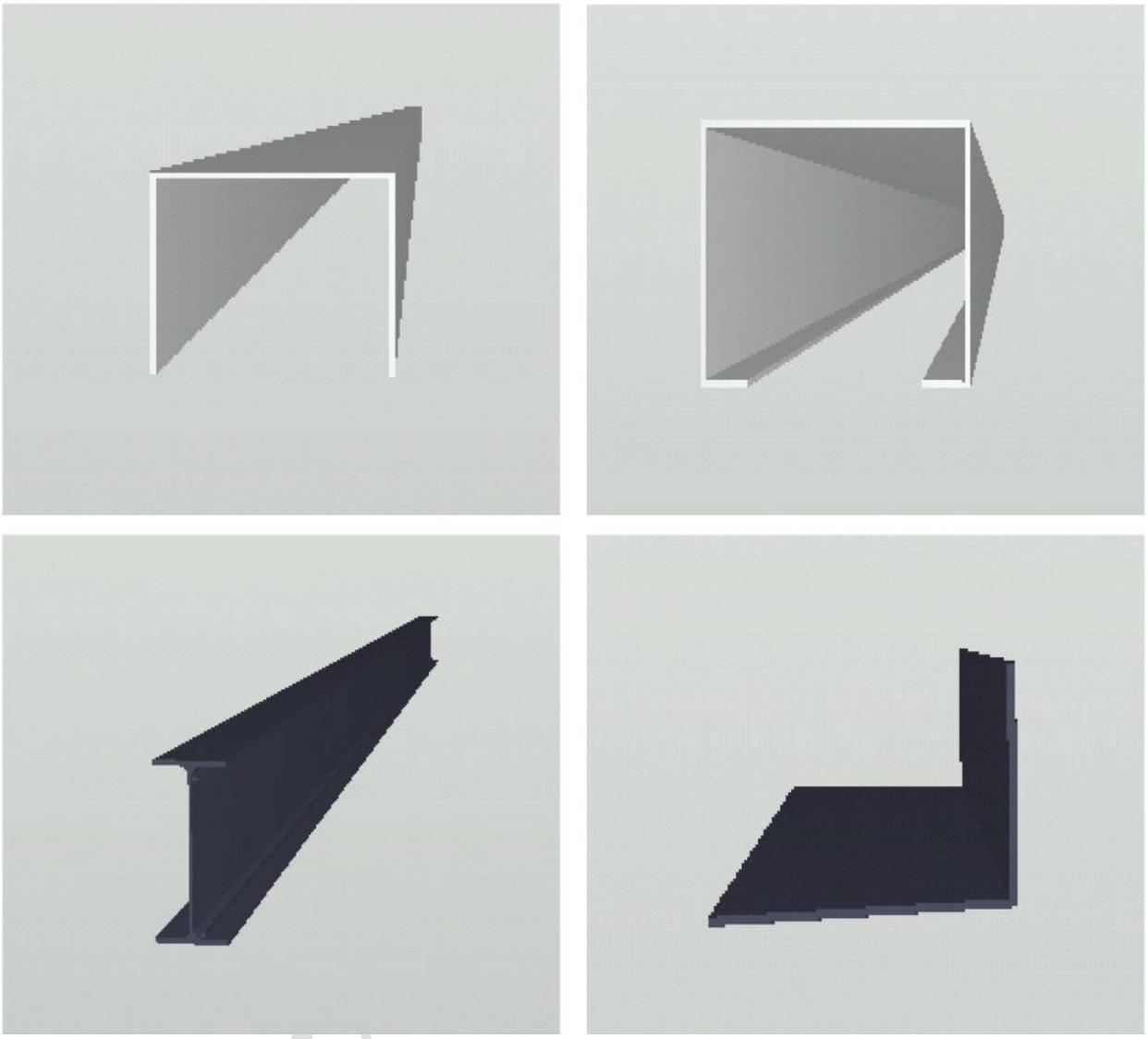

**Fig. 11.** U-Beam, C-Beam, I-Beam, and L-Beam

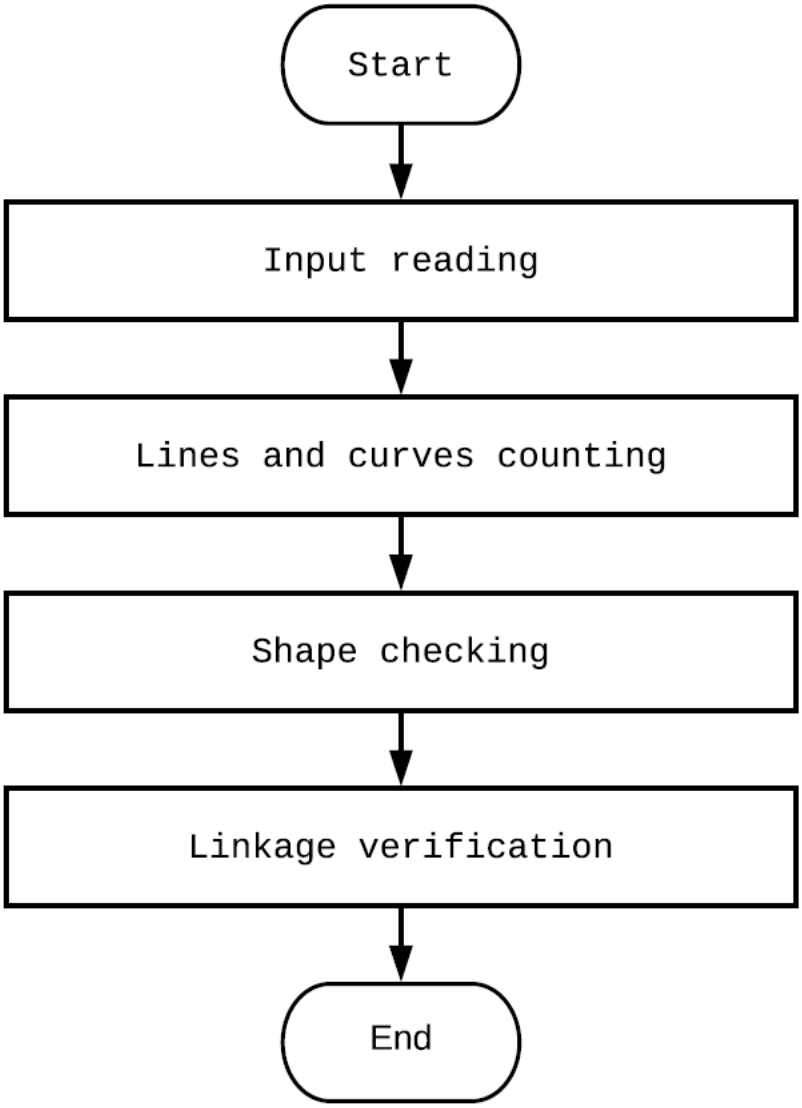


**Fig. 12.** Beam classification method

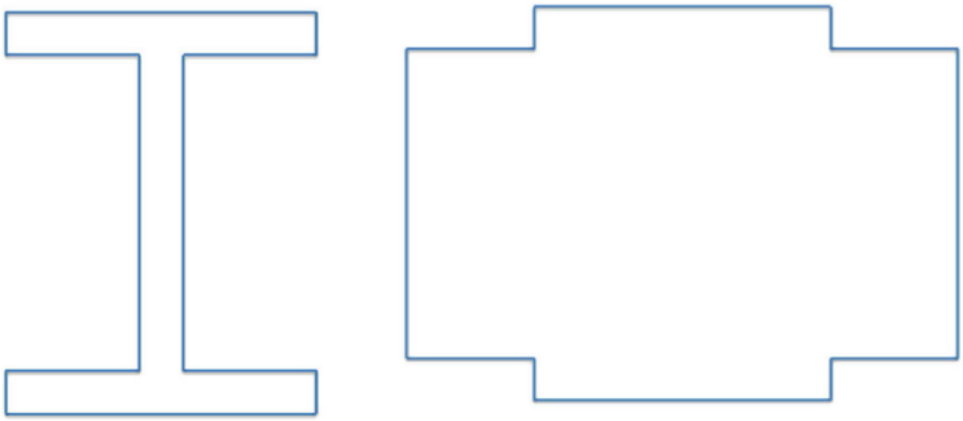

**Fig. 13.** Shapes with 12 lines but different line connection types

**Fig. 14.** The visualization of a truss

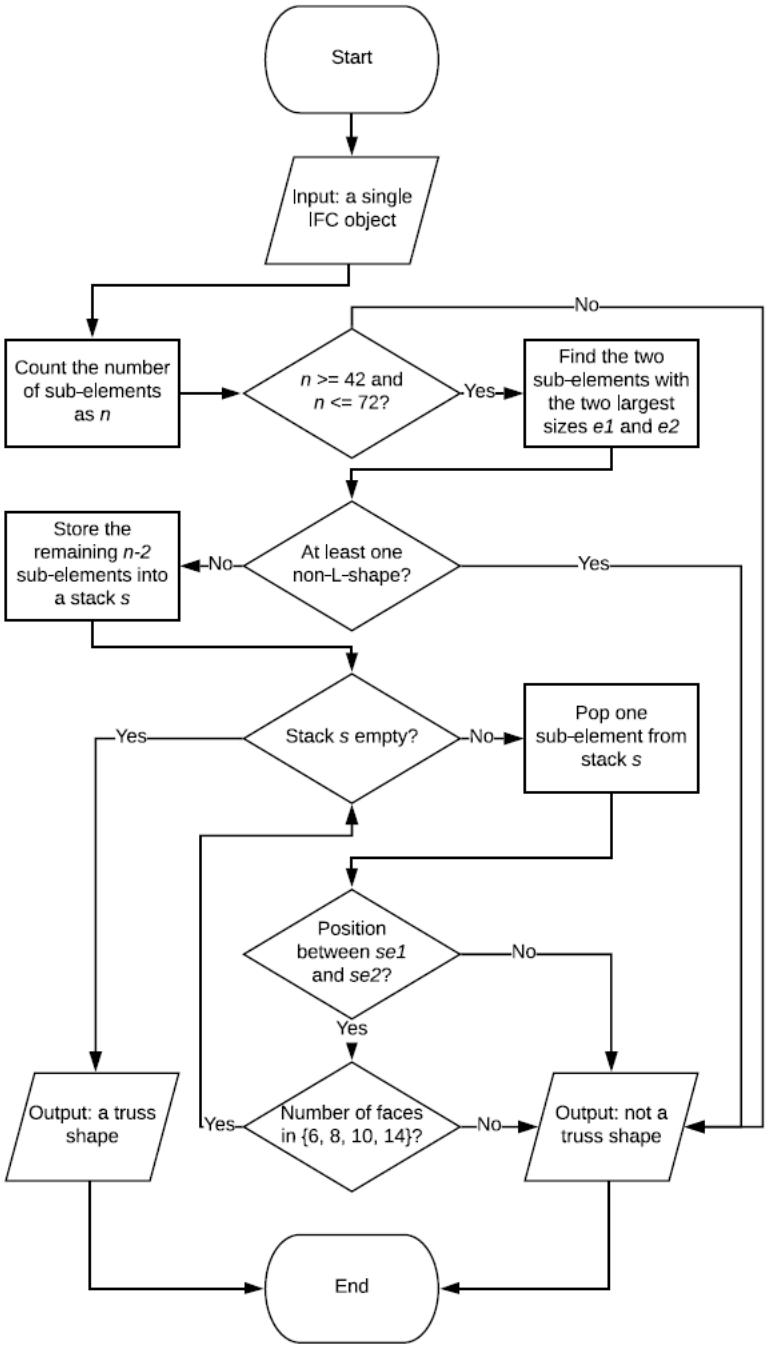


**Fig. 15.** The developed sub-algorithm for recognizing a truss

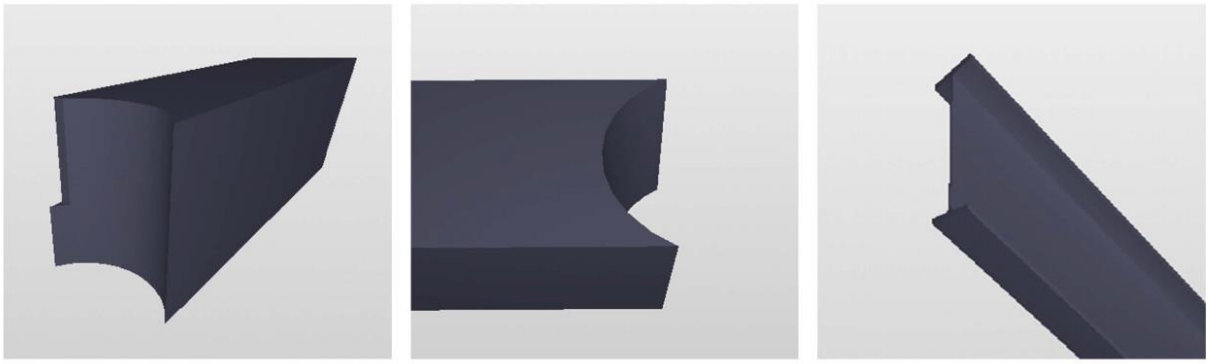

**Fig. 16.** Other shapes of beams

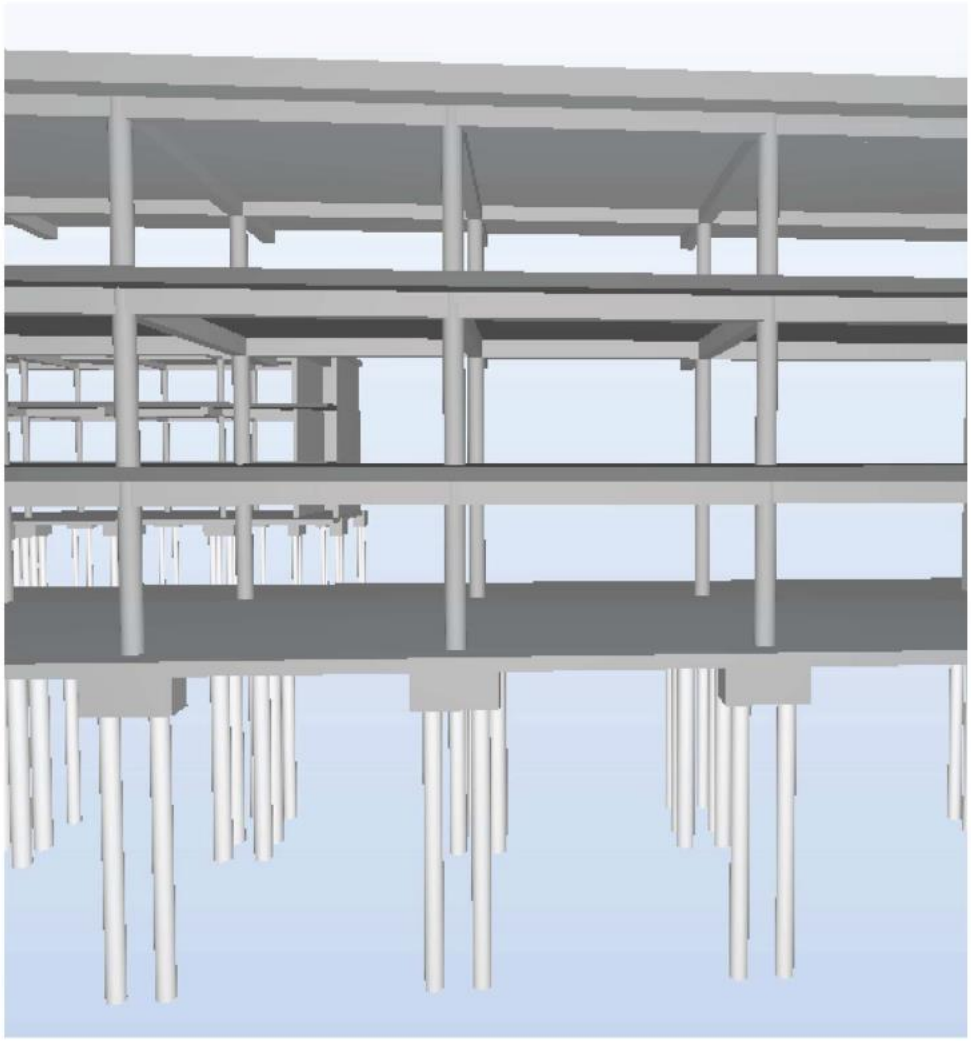

**Fig. 17.** Visualization of incorrectly classified objects

**Table 1.** Extracted Objects from IFC Models

| IFC model | Number of objects extracted |
|---|---|
| *Deplex_A* | 86 |
| *Rac_basic* | 85 |
| *Rst_advanced* | 883 |
| *Rst_basic* | 446 |
| *Tech_school* | 386 |
| *BIM_UK* | 5 |
| *Total* | *1,891* |

**Table 2.** Inter-Annotator Agreements of IFC Objects Manual Labeling

| Annotator | A | B | C | Average |
|---|---|---|---|---|
| A | - | 81.37% -> 99.79% | 81.52% -> 98.79% | 81.45% -> 99.29% |
| B | 81.37% -> 99.79% | - | 98.74% -> 98.58% | 90.06% -> 98.19% |
| C | 81.52% -> 98.79% | 98.74% -> 98.58% | - | 90.13% -> 99.69% |
| Average | 81.45% -> 99.29% | 90.06% -> 99.19% | 90.13% -> 98.69% | 87.21% -> 99.06% |

**Table 3.** Sample Majority Vote Labels

| AEC Object and Model Origin | Label by A | Label by B | Label by C | Majority Vote |
|---|---|---|---|---|
| IfcWall34 from *Duplex_A* | Wall | Wall | Beam | Wall |
| IfcWall35 from *Duplex_A* | Wall | Wall | Beam | Wall |
| IfcBeam132 from *Rst_advanced* | Beam | Colum | Beam | Beam |
| IfcBeam133 from *Rst_advanced* | Beam | Colum | Beam | Beam |
| IfcBeam133 from *Rst_advanced* | Beam | Colum | Beam | Beam |

**Table 4.** Manual Labels of Objects

| Labels | Number of objects labeled |
|---|---|
| Beams | 795 |
| Columns | 412 |
| Footings | 348 |
| Slabs | 74 |
| Walls | 262 |
| *Total* | *1,891* |

**Table 5**. Learning Curve Parameters

| Stage Number | Geometric Content of Study | Number of Correctly Classified Objects |
|---|---|---|
| Stage 1 | Rectangular shapes | 146 |
| Stage 2 | I-Beam with *IfcArbitraryClosedProfileDef* | 153 |
| Stage 3 | Slabs with *IfcArbitraryClosedProfileDef* | 184 |
| Stage 4 | *IfcCircleProfileDef* | 364 |
| Stage 5 | Four other built-in shape profiles | 538 |
| Stage 6 | *IfcCircleHollowProfileDef* | 554 |
| Stage 7 | *IfcCompositeCurve* | 557 |
| Stage 8 | *IfcClosedShell* | 637 |
| Stage 9 | Brep | 902 |
| Stage 10 | Clipping | 1,004 |
| Stage 11 | CSG | 1,005 |
| Stage 12 | Mapped Representation | 1,325 |
| Total | All | 1,325 |

**Table 6**. Possible Number of Lines and Curves for Rectangular Beam, U-Beam, C-Beam, I-Beam, and L-Beam

| Beam Types | Lines | Curves |
|---|---|---|
| Rectangular Beam | 4 | 0 |
| U-Beam | 8 | 0 |
| | 8 | 2 |
| | 8 | 4 |
| C-Beam | 12 | 0 |
| | 12 | 4 |
| | 12 | 8 |
| I-Beam | 12 | 0 |
| | 12 | 4 |
| L-Beam | 6 | 0 |
| | 6 | 1 |
| | 6 | 2 |

**Table 7**. Possible Beam Type According to Their Number of Lines and Curves in Geometric Representation

| Number of Lines | Number of Curves | Possible Beam Shape |
|---|---|---|
| 4 | 0 | Rectangular Beam |
| 6 | 0, 1, 2 | L-Beam |
| 8 | 0, 2, 4 | U-Beam |
| 12 | 0, 4 | I-Beam, C-Beam |
| 12 | 8 | C-Beam |

**Table 8**. BIM Object Classification Results of Training Data

| Object Types | Number of Actual Objects (*a*) | Number of Objects Classified into the Category (*b*) | Number of Correctly Classified Objects (*c*) | Recall (*c*/*a*) | Precision (*c*/*b*) |
|---|---|---|---|---|---|
| Beam | 561 | 561 | 561 | 100% | 100% |
| Column | 285 | 452 | 285 | 100% | 63.05% |
| Footing | 243 | 76 | 76 | 31.28% | 100% |
| Slab | 52 | 52 | 52 | 100% | 100% |
| Wall | 184 | 184 | 184 | 100% | 100% |
| *Total* | *1,325* | *1,325* | *1,158* | *87.40%* | *87.40%* |

**Table 9**. Beams Classification Results of Training Data

| Beam Shapes | Number of Actual Objects (*a*) | Number of Objects Classified into the Category (*b*) | Number of Correctly Classified Objects (*c*) | Recall (*c*/*a*) | Precision (*c*/*b*) |
|---|---|---|---|---|---|
| Rectangular Beam | 155 | 155 | 155 | 100% | 100% |
| C-Beam | 70 | 70 | 70 | 100% | 100% |
| I-Beam | 35 | 35 | 35 | 100% | 100% |
| L-Beam | 1 | 1 | 1 | 100% | 100% |
| U-Beam | 5 | 5 | 5 | 100% | 100% |
| Rectangular Beam with Cuts | 222 | 222 | 222 | 100% | 100% |
| Round Beam | 11 | 11 | 11 | 100% | 100% |
| Truss | 35 | 35 | 35 | 100% | 100% |
| Hollow Round Beam | 12 | 12 | 12 | 100% | 100% |
| Skewed I-Beam | 12 | 12 | 12 | 100% | 100% |
| *Total* | *558* | *558* | *558* | *100%* | *100%* |

**Table 10**. BIM Object Classification Results of Testing Data

| Object Types | Number of Actual Objects (*a*) | Number of Objects Classified into the Category (*b*) | Number of Correctly Classified Objects (*c*) | Recall (*c*/*a*) | Precision (*c*/*b*) |
|---|---|---|---|---|---|
| Beam | 234 | 234 | 234 | 100% | 100% |
| Column | 127 | 210 | 127 | 100% | 60.48% |
| Footing | 105 | 22 | 22 | 20.95% | 100% |
| Slab | 22 | 20 | 20 | 90.91% | 90.91% |
| Wall | 78 | 75 | 75 | 96.15% | 96.15% |
| *Total* | *566* | *561* | *478* | *84.45%* | *85.20%* |

**Table 11**. Beams Classification Results of Testing Data

| Beam Types | Number of Actual Objects (*a*) | Number of Objects Classified into the Category (*b*) | Number of Correctly Classified Objects (*c*) | Recall (*c*/*a*) | Precision (*c*/*b*) |
|---|---|---|---|---|---|
| Rectangular Beam | 64 | 64 | 64 | 100% | 100% |
| C-Beam | 29 | 29 | 29 | 100% | 100% |
| I-Beam | 9 | 9 | 9 | 100% | 100% |
| L-Beam | 0 | 0 | 0 | 100% | 100% |
| U-Beam | 3 | 3 | 3 | 100% | 100% |
| Rectangular Beam with Cuts | 99 | 99 | 99 | 100% | 100% |
| Round Beam | 5 | 5 | 5 | 100% | 100% |
| Truss | 15 | 15 | 15 | 100% | 100% |
| Hollow Round Beam | 8 | 8 | 8 | 100%<br>100%<br>100% | 100%<br>100%<br>100% |
| Skewed I-Beam | 2 | 2 | 2 | 100% | 100% |
| *Total* | *234* | *234* | *234* | *100%* | *100%* |

**Table 12**. Time Complexity of the Proposed Method in Comparison with the State-of-the-Art Methods

| Methods | Time Complexity |
|---|---|
| Proposed Method | O(1), constant time. |
| Ma et al. 2017 | O(kn), k is the highest number of properties for a studied object, and n is the number of studied objects |
| Sacks et al. 2017 | O(n) in theory. May be higher in practice. |