INVARIANT SIGNATURES FOR SUPPORTING BIM INTEROPERABILITY

by

Jin Wu

A Dissertation

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Doctor of Philosophy

School of Construction Management Technology

West Lafayette, Indiana

August 2021

ACKNOWLEDGMENTS

First and foremost, I would like to express my profound gratitude to my advisor Professor Jiansong Zhang for his guidance and support. It has been an honor to be his first direct Ph.D. student. I am deeply grateful for all his time and effort to help me learn and grow to be responsible, capable, and productive. I also want to thank my committee members Professor Luciana Debs, Professor Yi Jiang, Professor Randy R Rapp, and Professor Hazar Nicholas Dib for their insightful advice, precious time, and constructive comments.

I would like to give special thanks to my love, Xuan Zhang, for being supportive and patient during my Ph.D. study. I would also like to offer my special thanks to my parents and my sisters for their encouragement, trust, and unconditional support.

Thanks to everyone from our research lab, the Automation and Intelligent Construction (AutoIC) Lab. They have enriched both my professional and personal life at Purdue University. My thanks also go to my friends at the Purdue University.

I gratefully acknowledge the National Science Foundation (NSF) and the Purdue University that generously supported/sponsored my research. This material is based on work supported by the NSF under Grants No. 1745374 and No. 1827733. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author and do not necessarily reflect the views of the NSF.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

DEFINITIONS

Automated compliance checking (ACC) of building codes. Review and analyze the building design to check if it meets the requirements documented in building codes in an automated way to simplify the process (Zhang and El-Gohary 2016c).

Architecture, engineering, and construction (AEC). A collective term for the three industries.

Building information modeling (BIM). “A data-rich digital representation cataloging the physical and functional characteristics of design and construction” (GSA 2007).

Industry Foundation Classes (IFC). An open and neutral objected-based data format for AEC objects (BuildingSMART 2018a).

ABSTRACT

Building Information Modeling (BIM) serves as an important media in supporting automation in the architecture, engineering, and construction (AEC) domain. However, with its fast development by different software companies in different applications, data exchange became labor-intensive, costly, and error-prone, which is known as the problem of interoperability. Industry foundation classes (IFC) are widely accepted to be the future of BIM in solving the challenge of BIM interoperability. However, there are practical limitations of the IFC standards, e.g., IFC's flexibility creates space for misuses of IFC entities. This incorrect semantic information of an object can cause severe problems to downstream uses. To address this problem, the author proposed to use the concept of invariant signatures, which are a new set of features that capture the essence of an AEC object. Based on invariant signatures, the author proposed a rule-based method and a machine learning method for BIM-based AEC object classification, which can be used to detect potential misuses automatically. Detailed categories for beams were tested to have error-free performance. The best performing algorithm developed by the methods achieved 99.6% precision and 99.6% recall in the general building object classification. To promote automation and further improve the interoperability of BIM tasks, the author adopted invariant signature-based object classification in quantity takeoff (QTO), structural analysis, and model validation for automated building code compliance checking (ACC). Automation in such BIM tasks was enabled with high accuracy.

CHAPTER 1 – INTRODUCTION

Significant portions of this chapter can be found in:

"Automated BIM object classification to support BIM interoperability." In *Proc.*, *Construction Research Congress*, 706-715. DOI: 10.1061/9780784481301.070.

"New automated BIM object classification method to support BIM interoperability." in *Journal of Computing in Civil Engineering*, 33(5). DOI: 10.1061/(ASCE)CP.1943-5487.0000858.

"Invariant signatures of architecture, engineering, and construction objects to support BIM interoperability between architectural design and structural analysis." in *Journal of Construction Engineering and Management,* 147(1). DOI: 10.1061/(ASCE)CO.1943-7862.0001943.

1.1 Motivation and Overview

Building Information Modeling (BIM) is designed to serve different stakeholders at different life cycle phases, such as architects at the design phase and contractors at the construction phase (Eastman et al. 2011). BIM eases the analysis and control of a project prior to its construction. It facilitates the collaboration of different stakeholders by providing a platform with uniform standards and data in the architecture, engineering, and construction (AEC) domain. For such support, the data of building information models (BIMs) must contain accurate information in their digital representations and is easily interoperable for different stakeholders.

However, as the development of BIM was mainly carried out by proprietary software providers, the BIM companies used different data structures and data formats. As a result, data transfer between different BIM software turned out to be labor-intensive and costly because of the needed manual efforts in converting data between different formats and fixing information inconsistency or adding the missing information. This data exchange process is also error-prone because of the labor involvement. A small error in the BIM data may result in malfunctions in the construction processes such as material misuse or dimension mismatch, therefore resulting in costly construction failure and/or rework. The difficulty of the data exchange is known as the problem of interoperability.

According to a conservative estimate by the National Institute of Standards and Technology (NIST) (Gallaher et al. 2004), the lack of interoperability in the U.S. capital facilities industry costs \$15.8 billion per year. Industry Foundation Classes (IFC) is widely believed to be the future of BIM because the IFC data format is open and neutral (BuildingSMART 2018a). However, there are practical limitations of the IFC standards, especially the entity misuse created by IFC's flexibility. To prevent such error from happening, it is critical to generate flawless data at each phase of the construction, which requires an effective checking of the model in different phases. One of the key challenges in such a checking process is the labeling of AEC building objects in BIM, i.e., annotating the objects with their correct categories. Although some efforts have been conducted to develop methods that can automatically label BIM data (Ma et al. 2018, Koo et al. 2019), the results of automated labeling still need to be manually checked in practical use, to ensure the quality of the labels. To achieve full automation of such labeling process, the BIM object classification method still needs further research and development to cover more generic building elements with high accuracy.

In this dissertation, the author proposed to use invariant signatures, which are a set of features that captures the intrinsic properties of AEC objects. The invariant signatures were an effort to solve interoperability by providing automated AEC object classification with high accuracy. The object classification can be used for many BIM tasks, such as quantity takeoff (QTO), structural analysis, and automated building code compliance checking (ACC).

1.2 State of the Art in BIM Interoperability and Knowledge Gaps

Object classification distinguishes BIM from 3D computer-aided design (CAD) by carrying the semantic information of objects in a building model (Ma et al. 2018). Ma et al. (2018) proposed an integrated approach to classify AEC objects that combined domain knowledge of geometric features and pairwise relationships of 3D objects into a tailored matching algorithm. Their algorithm can process various complex 3D geometries and compile a knowledge base in civil engineering. In addition, in their experiment, Ma et al. (2018) achieved 100% accuracy in their two bridge models and provided a knowledge matrix of the objects.

Distinct from the rule-based approaches taken by Ma et al. (2018), Koo et al. (2019) proposed a classification method using support vector machines (SVM), a machine learning algorithm commonly used for classification. They proposed a feature set that could map IFC objects to selected IFC classes with an average F1-score of 94.9% for eight classes. SVM is one of the most robust supervised machine learning algorithms.

Although significant results have been achieved for object classification of BIM objects, the practical gap has not been filled, because (1) the lack of interoperable features for AEC objects, (2) limited accuracy for AEC object classification, with more advanced features and better machine learning algorithms needed, (3) the practical gaps for solving interoperability in BIM tasks, e.g., model exchange for QTO and structural analysis, information extension for ACC.

1.3 Proposed Approach

In this dissertation, a new set of features of BIM-based AEC objects, namely, the invariant signatures, are defined, extracted, and used in object classification. Then the invariant signatures and the object classification are used together to support practical BIM tasks, including QTO, structural analysis, and ACC. To support the experiment, a new dataset was collected, manually labeled, and shared.

1.3.1 Introduce a new BIM-based AEC object dataset

The author built a dataset that covers a wide range of different shapes and representations of IFC objects. Each type has hundreds of unique AEC objects.

For object labeling, each object is labeled by independent annotators with inter-annotator agreements assessed. For instances that the annotators do not agree on, majority votes are used to label the data.

With the labeled entities, the dataset can be used in later experiments. The data is made public and shared through the Purdue University Research Repository (PURR) (Wu and Zhang 2018b, Wu and Zhang 2021b). The data can be used to reproduce the results and conduct further research by other researchers.

1.3.2 Construct invariant signatures

Invariant signatures are a set of properties that capture the geometric nature and intrinsic relations of an AEC object that do not change with data schema, software implementation, or language/cultural contexts (Wu et al. 2021).

The author proposes to construct invariant signatures in three sub-types: geometric signatures, locational signatures, and metadata signatures. Geometric signatures capture the geometric information of common shapes, such as rectangles and cylinders. Locational signatures capture the position information of an object, including the absolute position, relative position, and orientation. Metadata signatures capture representation-level information, especially the representation used in IFC, e.g., the average number of vertices among faces in a boundary representation (Brep).

Invariant signatures are extracted iteratively in BIM applications. For each BIM task, a subset of the invariant signatures is used to represent the AEC objects in the model. For example, QTO only uses the geometric signatures, while structural analysis uses both geometric and locational signatures.

1.3.3 BIM-based AEC object classification

With the invariant signature-based features of AEC objects, two approaches are developed for AEC object classification, namely, a rule-based approach and a machine learning (ML) approach. The two methods are developed independently on the same dataset, and the results are compared in terms of precision and recall.

1.3.4 Applications of invariant signature and object classification

The invariant signatures have broader use in addition to object classification. As an illustration, the invariant signatures and the classification algorithms are used together to support QTO, structural analysis, and ACC.

1.4 Problem Statement

Interoperability is a costly problem in BIM because of the need for manual efforts in information checking, reentry, correction, etc., which is time-consuming, costly, and error-prone. Automation in such tasks is expected to reduce time, cost, and error. While significant efforts have been put into addressing BIM interoperability, the previous efforts are still limited, because they (1) still require significant manual effort in BIM tasks, (2) lack an intuitive uniform interoperable representation that can be used for different BIM tasks, and (3) have not achieved seamless and universal interoperability.

1.5 Research Objectives and Questions

The objective of this research is to extract uniform features from AEC objects, and use these features in BIM tasks, including BIM object classification, structural analysis, QTO, and ACC.

Objective 1. Build a new uniform and open AEC object dataset

Collect, label, and prepare a dataset of AEC objects from IFC models with good coverage in object types, object sizes, and data representations.

Research Questions: What are reliable sources for collecting the data? How to develop gold standard that can be widely accepted? What are the requirements for coverage of building object representations? How to share the data to promote collaboration and communication?

Objective 2. Construct invariant signatures of AEC object

Define, construct, and test invariant signatures of AEC objects from the collected dataset that can fully represent the object using a data-driven approach.

Research Questions: What properties the invariant signatures should have? What advantage do the invariant signatures have over other methods or concepts?

Objective 3. Apply invariant signatures in AEC object classification

Apply invariant signatures in AEC object classification to achieve high precision and recall. Develop algorithms using both a rule-based method and a machine learning method.

Research Questions: How to achieve high precision and recall for object classification? How to balance between recall and precision? Between machine learning and rule-based, how do they compare in the AEC object classification task? What are the advantages and limitations of both methods?

Objective 4. Apply invariant signatures and object classification to practical BIM applications

Apply invariant signatures and object classification in QTO, structural analysis, and ACC.

Research Questions: What is the performance comparing to the state of the art? What are the advantages of using invariant signatures in terms of efficiency, cost, and precision? What other BIM tasks can be improved using invariant signatures and object classification?

.

1.6 Significance of the Problem

BIM object classification remains a key challenge in the development of full automation for BIM-based applications. The author seeks to address this challenge by automated classification of BIM objects with high accuracy and efficiency.

For such support of automation, the data used in BIM must contain accurate information. A small error in BIM data can result in malfunctions in the construction process, such as material misuse and size mismatch, therefore resulting in a construction failure. In addition, it is costly to correct the error in later phases of the construction. To generate flawless data, it is essential to check the model during different phases of the construction to conduct correct QTO and structural analysis. ACC also requires accurate information extraction and mapping, which checks the safety, consistency to allow the model to function safely and efficiently. For full automation in BIM applications, annotating the objects in BIMs with correct categories is a key premise. Then practical BIM applications can be developed, potentially with full automation. With a fully automated classifier and fully automated BIM applications, efforts dedicated to solving interoperability issues manually can be reduced or eliminated. The author's method facilitates the automation of the AEC domain, provides insights for the understanding of BIM, and promotes interoperability for BIM applications, thus contributes to the body of knowledge in the AEC domain.

1.6.1 Intellectual merit

The research has the following intellectual merit.

- Propose a set of features (invariant signatures) that capture the geometric essence of BIM-based AEC objects. The invariant signatures are interoperable and can be used in QTO, structural analysis, and ACC.
- A new iterative method for developing rule-based algorithms for AEC object classification. The algorithms relied on invariant signatures, i.e., the inherent geometric features, of AEC objects rather than entity or attribute names and therefore prevented classification errors caused by misuse of entities, which are stable and reliable.
- A new generic method for developing machine learning algorithms for AEC object classification. The method can systematically select features and the best-fit machine learning algorithms.

1.6.2 Broader impact

The research has the following broader impact.

- A new open dataset for research. The dataset can be used to reproduce the results and conduct further research in BIM interoperability and BIM-based AEC task automation for other researchers.
- Improve the accuracy for object classification, thus preventing misuses of IFC entities. Automating the BIM-based AEC object classification is expected to save time, save efforts, and reduce the errors of object classification by eliminating human-caused errors that may occur during the manual annotating process.
- Improve interoperability of QTO and structural analysis. Promote automation by developing algorithms for data exchange from IFC-based BIM models to structural analysis software.
- Improved automation in model validation by extending the extracted information with logic-based inference. Develop heuristic algorithms that automate the otherwise manual effort dedicated to checking missing information, inferring non-explicit information, and mapping to building code concepts.
- Save time and manual effort, reduce human-caused errors on BIM tasks, including AEC object annotation, data exchange, and model validation.

1.7 Assumptions

To allow the algorithms to work in different models, the data in those testing models must be consistent with those of the training models. The developed algorithms may not achieve the same level of performance if the testing model contains totally different patterns from the training data, i.e., if a certain pattern cannot be found in the training data, then the developed algorithms may not generate expected results. As a premise, a key assumption is that the training and testing data contain the same representation patterns. In this dissertation, this condition is satisfied to the largest extent by randomly dividing the data into training set and testing set.

In addition to the data consistency, another assumption is that the framework and algorithms can be used in a broader application setting. The author uses 1,900 object instances in AEC object classification, two models for QTO, ten models for structural analysis, and five models for ACC. The experiment serves as an illustration of the proposed framework. The algorithms serve as proof-of-concept to illustrate the potential of such applications.

1.8 Limitations

1.8.1 Dataset

As a first step towards building a comprehensive BIM object classifier, the author establishes a new dataset tailored for object classification. The dataset contains five BIM models with 1,900 object instances in IFC format. The dataset provides verified labels and processed features for the objects in a systematic way. While the data is sufficient for developing proof-of-concept algorithms, the dataset was still small compared to the dataset in other domains, such as the ADE20K (Zhou et al. 2018) that contained 22,210 images for image processing. The dataset needs to be continuously developed to contain more models to cover more robust object representations.

1.8.2 Object classification

In spite of the promising experimental results and the ability to achieve 100% recall and precision in automated AEC object classification for certain object categories, the following limitations of the proposed method are acknowledged. First, the proposed rule-based method for

AEC object classification is labor intensive in the algorithm development phase, especially when a comprehensive algorithm is pursued to cover all possible geometric shape representations of AEC objects.

Second, a universal classifier requires a large amount of data in training for all the possible configurations. Instead of the full development of the universal classifier, the author focuses on the method of developing such classifiers to show the potential of this proposed method using a data-driven approach. While the methods are expected to be adaptable to more object types with data of that type, practical obstacles may occur in the actual development.

1.8.3 QTO and structural analysis

The author acknowledges two main limitations of the QTO and structural analysis methods as follows. First, the invariant signatures are only tested for regular-shaped buildings and members, for both QTO and structural analysis. For example, it was not tested for curved or irregular shapes. Second, the experiment of structural analysis only used one structural analysis software. The algorithm development needs to be expanded to cover more complicated structures such as high-rise buildings, incorporate curved and irregular-shaped building elements into consideration, and test the invariant signatures on more software platforms.

1.8.4 Model validation

The author acknowledged the following limitations of the proposed model validation method in this dissertation: (1) the proposed method was tested in only one chapter (i.e., Chapter 10) of the IBC 2015, how it will perform in other chapters and other building codes remain to be tested. However, the author would expect comparable results on other chapters and other building codes using the same method. (2) In order for the processing of inferable concepts to be error-free, the assumption used in the heuristic rules needs to be broadly applicable in any foreseeable BIMs. (3) Due to the nature of rule-based algorithms, it is time-consuming and labor intensive to grow the number of concepts and patterns covered, whereas error is always possible before the number of concepts and patterns saturate. However, the author believes the number of concepts and patterns in this context is enumerable, and a compatible set of rules are feasible with careful implementation, such as by performing a comprehensive check after each new rule is added. In addition, while the theoretical optimal rule set may be hard to achieve, it is always feasible to arrive at practically “comprehensive” solutions for a known scope (e.g., concepts in International Fire Code) using data-driven approaches.

1.9 Delimitations

All the methods and algorithms are developed for as-design models. As-built models, such as point-cloud data, are not in the scope of this dissertation.

The collected data are either IFC models or are converted to IFC models, because IFC standards are open and neutral. Other types of BIM models are not directly covered in this dissertation. They are converted into IFC models with existing methods and software. Errors such as entity misuse are allowed in the converted models.

For AEC object classification, the author focuses on five main types of building components, which are beams, columns, footings, slabs, and walls. Sub-types of beams are also included. Other types of AEC objects, e.g., framed doors and swing doors, are not tested in this dissertation. However, the methods are expected to work for developing algorithms for those other entities as well. As an illustration, the author developed algorithms for distinguishing interior doors from exteriors doors, following the Uniformat (Uniformat 2010).

For the machine learning method of AEC object classification, the author uses traditional machine learning algorithms. Based on the size of the collected data, which is at the scale of thousands of items, the author excluded the option of deep learning, which is under fast development and achieved many promising results but requires a significantly larger size of data.

1.10 Article-Based Dissertation Statement

As an article-based dissertation, the author used four journal publications (Wu and Zhang 2019b, Wu et al. 2021a, Wu et al. 2021b, Wu and Zhang 2021a) and two conference publications (Wu and Zhang 2018a, Wu and Zhang 2019a).

The author started the research on object classification using rule-based algorithm (Wu and Zhang 2018a, Wu and Zhang 2019b). The data was then used to create a public dataset and to generate invariant signatures (Wu and Zhang 2019a). The invariant signatures are used in structural analysis (Wu et al. 2021a), machine learning-based object classification (Wu et al 2021b), and model validation (Wu and Zhang 2021a).

The *Construction Research Congress* provided permission for me to publish this article titled "Automated BIM object classification to support BIM interoperability." in this dissertation.

The *2019 ASCE International Conference on Computing in Civil Engineering* provided permission for me to publish this article titled "Introducing geometric signatures of architecture, engineering, and construction objects and a new BIM dataset." in this dissertation.

The *Journal of Computing in Civil Engineering* provided permission for me to publish this article titled "New automated BIM object classification method to support BIM interoperability." in this dissertation.

The *Journal of Construction Engineering and Management* provided permission for me to publish this article titled "Invariant signatures of architecture, engineering, and construction objects to support BIM interoperability between architectural design and structural analysis." in this dissertation.

The *Journal of Computing in Civil Engineering* provided permission for me to publish this article titled "Constructing invariant signatures for AEC objects to support BIM-based analysis automation through object classification." in this dissertation.

The *Journal of Computing in Civil Engineering* provided permission for me to publish this article titled "Model validation using invariant signatures and logic-based reasoning for automated building code compliance checking." in this dissertation.

1.11 Summary

In summary, behind all the practical gaps in BIM tasks and applications is the lack of intuitive, uniform, and interoperable representations that can be seamlessly used for different BIM tasks. The author proposes to use invariant signatures of AEC objects as a solution, which are "a set of intrinsic properties of the object that distinguish it from others and that do not change with data schema, software implementation, modeling decisions, and/or language/cultural contexts" (Wu et al. 2021). The invariant signatures are able to provide interoperable representations for BIM and AEC objects. With this theoretical foundation, practical BIM tasks can be automated seamlessly. For each BIM task, a new method is proposed for its automation with the adoption of invariant signatures. For the development of invariant signatures and the applications, Fig. 1 shows the timeline of the development and articles.

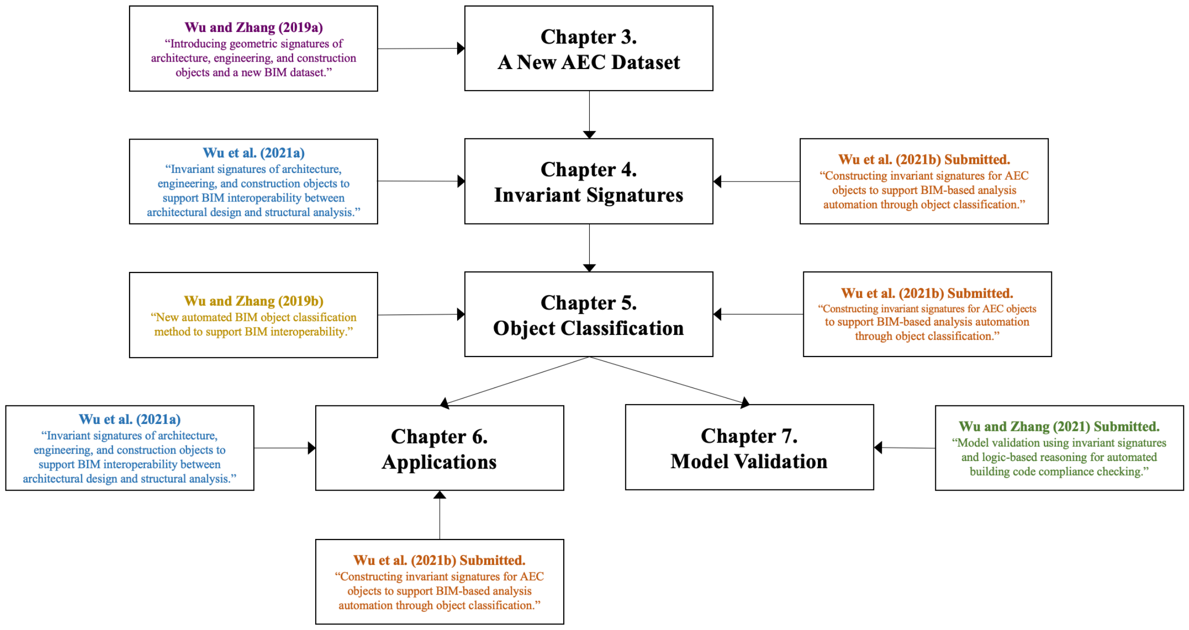

Wu and Zhang (2019a)
"Introducing geometric signatures of architecture, engineering, and construction objects and a new BIM dataset."
Chapter 3. A New AEC Dataset
Wu et al. (2021a)
"Invariant signatures of architecture, engineering, and construction objects to support BIM interoperability between architectural design and structural analysis."
Chapter 4. Invariant Signatures
Wu et al. (2021b) Submitted.
"Constructing invariant signatures for AEC objects to support BIM-based analysis automation through object classification."
Wu and Zhang (2019b)
"New automated BIM object classification method to support BIM interoperability."
Chapter 5. Object Classification
Wu et al. (2021b) Submitted.
"Constructing invariant signatures for AEC objects to support BIM-based analysis automation through object classification."
Wu et al. (2021a)
"Invariant signatures of architecture, engineering, and construction objects to support BIM interoperability between architectural design and structural analysis."
Chapter 6. Applications
Chapter 7. Model Validation
Wu and Zhang (2021) Submitted.
"Model validation using invariant signatures and logic-based reasoning for automated building code compliance checking."
Wu et al. (2021b) Submitted.
"Constructing invariant signatures for AEC objects to support BIM-based analysis automation through object classification."

Fig. 1. Visualization of chapters flow with corresponding publications.

CHAPTER 2 – LITERATURE REVIEW

Significant portions of this chapter can be found in:

"Automated BIM object classification to support BIM interoperability." In *Proc., Construction Research Congress*, 706-715. DOI: 10.1061/9780784481301.070.

"New automated BIM object classification method to support BIM interoperability." in *Journal of Computing in Civil Engineering*, 33(5). DOI: 10.1061/(ASCE)CP.1943-5487.0000858.

"Invariant signatures of architecture, engineering, and construction objects to support BIM interoperability between architectural design and structural analysis." in *Journal of Construction Engineering and Management,* 147(1). DOI: 10.1061/(ASCE)CO.1943-7862.0001943.

"Constructing invariant signatures for AEC objects to support BIM-based analysis automation through object classification." in *Journal of Computing in Civil Engineering*, submitted.

"Model validation using invariant signatures and logic-based reasoning for automated building code compliance checking." in *Journal of Computing in Civil Engineering*, submitted.

"Introducing geometric signatures of architecture, engineering, and construction objects and a new BIM dataset." In *Proc., 2019 ASCE International Conference on Computing in Civil Engineering*, 264-271. DOI: 10.1061/9780784482421.034.

2.1 Building Information Modeling (BIM)

2.1.1 Building information modeling (BIM)

Building information modeling (BIM) is "a data-rich digital representation cataloging the physical and functional characteristics of design and construction" (GSA 2007). BIM has been gaining popularity since the 1980s when the first BIM software - the ArchiCAD's Radar CH – was developed (Quirk 2012). Recent BIM software, such as Autodesk Revit, Bentley AECOsim, Solibri Model Viewer, and BIMserver, serve primarily as architectural design and visualization tools (Eastman et al. 2011).

Using BIM technology, architecture, engineering, and construction (AEC) models can be built virtually with accurate digital representations prior to their physical construction. Comparing to a traditional manual process, BIM and related technologies allow better analysis and control of a project, such as in cost estimation (Akanbi et al. 2020) and progress control (Ren and Zhang 2021). These computer-generated models can support the construction, fabrication, and procurement activities of a construction project with accurate information in a digital format, which saves time and effort by supporting the automation of construction engineering and management tasks (Eastman et al. 2011, Azhar et al. 2011, Hussain and Choudhry 2013, Santos et al. 2017, Wong Chong and Zhang 2021).

Such pre-construction visualization improves collaboration between different stakeholders such as planners, designers, structural engineers, construction managers, and field workers. Better collaboration between these stakeholders will likely improve the 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 programs that a BIM user can choose from. For example, ETakeoff (2021) helps people with the cost estimation task, ANSYS (2021) helps people with the structural analysis task, and STR Vision (2021) provides a 4D (3D + time) and 5D (4D + cost) 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 options. Considering that for the same project the BIM data in these different programs belong to the same building structure, it will be a waste of opportunities to create BIM data from scratch in each of the software applications. 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 programs need one converter that converts the BIM model between the formats of the two, ten BIM software programs would require $C\binom{10}{2}$ = 45 converters (theoretically) to achieve interoperability between all of them. The number of converters increases quadratically with the number of BIM software programs, resulting in a high cost and inconvenience for achieving interoperability between all BIM software. To reduce the high cost of achieving BIM interoperability, 12 companies collaborated to develop a uniform standard for BIM, 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, the IFC schema still has major problems when transforming to other proprietary software formats and vice versa (Ma et al. 2006; Pazlar and Turk 2008).

2.1.2 Information exchange in building information models (BIMs)

Information exchange has been frequently studied (Sarawagi 2008, and Yang et al. 2019), especially using the ontology-based approach in recent research (Wimalasuriya and Dou 2010, Fernández et al. 2011, Paliouras et al. 2011, and Yehia et al. 2019). For information exchange in BIMs, in a design science research by Ding et al. (2017), a real-time quality checking system was accomplished using IFC-based Inspection Process Model (IFC-IPM). Their system demonstrated the efficiency in quality information exchange and could be used in inspection activities. Kim et al. (2016) developed a new approach to extract and process material information in BIMs, to explore energy-saving options early in the design phase. The major limitation of such an approach was the inaccuracies during simplifications in construction/material data. Their new system consisted of three parts, namely, information extraction, material property matching, and file writing. The system improved the efficiency of energy modeling by eliminating the manual information input and increased the accuracy using the developed matching algorithms.

For information exchange of IFC models in ACC, Zhang and El-Gohary (2019) proposed a machine learning-based approach to map building code concepts and relations to IFC elements and relations. Their method achieved 77% matching accuracy in IFC elements and 78% accuracy in IFC relations, which is representing the state of the art. For practical ACC, performance improvement will be needed. In addition, many concepts in building codes do not have a one-to-one mapping to IFC entities. The bridge between building code concepts and IFC entities still need to be built with additional ways.

2.2 IFC Schema

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 for common shapes than uncommon shapes.

IFC schema has been developed from IFC1.0 to IFC4.3 RC2 (buildingSMART 2021). Because of the delay in the wide adoption of the latest version, IFC2x3 was the most widely used version of the IFC schema at the time of this research. Based on the collected data, the author chose to use IFC2x3 in the experiment. There are 117 defined types and 653 entities, including 33 types of entities for building elements in the IFC2x3 schema (buildingSMART 2007). For example, *IfcBeam* is a building element entity that is used to define a beam instance. Fig. 2 provides the visualization of a beam in IFC. Fig. 3 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.

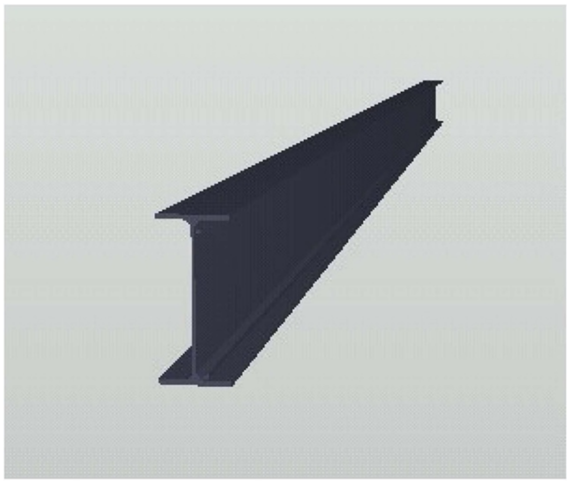
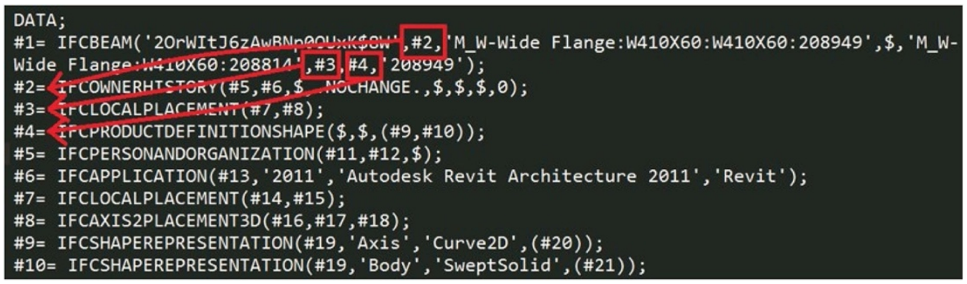


Fig. 2. Visualization of a wide flange beam/I-beam.


```
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. 3. IFC data of an I-beam as represented by an *IfcBeam*.

To track the information of an object, a reference-tracing algorithm is required (Won et al. 2013). IFC is widely believed to be the future of BIM because the IFC data format is open, platform neutral, and intended to be used by all disciplines and all life cycle phases of a project in the AEC domain (BuildingSMART 2018a). Created by the collaboration of twelve companies, IFC was designed to create a uniform standard to address the interoperability challenge (Hamil 2012). BuildingSMART has been constantly developing and refining IFC. As a result, it became one of the most promising attempts and is gaining more popularity. The ISO-registered IFC data standard facilitates BIM interoperability by allowing a "one-to-many" information flow between different BIM applications, which enables the mapping between one central model and representations in various applications, therefore brings flexibility into BIM representation. 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. However, such flexibility also creates challenges in data consistency. While the future is promising, the IFC schema still has a major flaw in its actual use when transforming data between different software formats (Ma et al. 2006; Pazlar and Turk 2008). An important factor in ensuring IFC data consistency and integrity is the correct object classification.

An IFC model usually consists of a plurality of AEC objects represented by IFC entities. Correct use of the IFC schema indicates the use of the correct entity that the object intends to represent. Although 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. 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.

In extreme cases, most of the objects in a model contain entity misuses. For example, in a bridge model with 59 objects in total, there were 35 objects (59%) with entity misuses (Ma et al. 2018). Due to the flexibility of IFC, one object can be represented by different IFC entities. For example, a wall can be represented by *IfcExtrudedAreaSolid* or *IfcFacetedBrep*, which is by extending a 2D plan or by defining all the 6 faces of a rectangular parallelepiped, respectively. Such an approach separates the IFC entities used with the geometric information it is attached to, which created a space for entity misuse.

To prevent such errors and negative consequences resulting from misuses of IFC entities, an automated checking process is required, to improve the current approach which is mainly labor-intensive manual checking. The checking process requires the accurate classification of an object by its content. As a result, new methods in AEC object classification are needed.

2.3 Shared Data for Research and Development

A Shared dataset (e.g., ImageNet, Flickr) not only provides people with resources for research, but also enhances the synergistic effect of research efforts from different teams by providing a common ground for comparison and discussion. It has been a common practice in computer science domains and helped advance research discoveries and technology development. For example, in the computer vision domain, Guillaumin et al. (2014) developed automated annotation methods for images using ImageNet. Yin et al. (2009) explored social tagging graph-based web object classification using Flickr. In the natural language processing domain, Reid et al. (2018) developed social science interpretation methods based on decompounded lexicon induction technics, through the use of a Consumer Complaint Database in their development.

Open datasets are the cornerstones of many research studies and provide a platform for comparison and collaboration. For example, in biology, Rezac et al. (2018) proposed a conformational method using MPCONF196 Benchmark Energy Data Set, which contained data that was carefully selected with both high and low-energy compounds. Rezac et al. (2018) selected cyclic and acyclic model peptides and other macrocycles to increase the accuracy of density functional theory (DFT)-based method for building the structure of complex biomolecules. Their method has shown statistically significant agreement with the ground truth. The method of Rezac et al. (2018) provides an efficient and accurate model that has the potential to help people fully understand the structural determinant of complex biomolecules, e.g., the information contained in the structure of DNA. In computer science, Zhou et al. (2017) developed a method for Scene Parsing through the use of the ADE20K Dataset (Zhou et al. 2018). They introduced and analyzed the ADE20K dataset, which contains diverse annotations of senses, objects, and parts of objects. Consistent annotations of the images were created following a labeling protocol, and the dataset was larger and more diverse compared to many other image datasets, such as COCO (Lin et al. 2014) and ImageNet (Russakovsky et al. 2015). The method proposed by Zhou et al. (2017) used a Cascade Segmentation Module to parse the images, that could remove image content and synthesize images automatically. Without the MPCONF196 dataset, the ADE20K dataset, and other related image datasets, such research developments would not have been possible or as successful.

2.4 Existing Methods for AEC Object Classification

2.4.1 General object classification

Object classification is the process in which objects are recognized, differentiated, and interpreted meaning that the objects are grouped for some specific purpose. Object classification in an automated fashion involves two essential steps: feature extraction (Langley 1994) and feature-based classification (Ullman 2007).

There is a lot of research on two-dimensional (2D) object classification to detect and classify objects from 2D images. For example, Cohen and Lefebvre (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 subparts 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 et al. 2015), bridge components (Narazaki et al. 2018), and cracks and defects in a structure (Feng et al. 2017; Gopalakrishnan et al. 2017), among others.

In contrast to 2D object classification, three-dimensional (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 (2000) 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). Red, green, and blue-depth (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, Socher et al. (2012) proposed a combination of convolutional and recursive neural networks (CNNs and RNNs) to classify 3D objects from RGB-D data, in which 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 object classification in RGB-D data that 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 aforementioned 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, an 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 compared to that of as-built models. In this domain, Qin et al. (2014) proposed a 3D computer-aided drafting (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 accuracy 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 buildings and apartment buildings. They achieved a cross-validation accuracy of 90.79% on 1,953 building objects.

2.4.2 Identification of building components

There is also no lack of research in automated identification of building components. For example, Quintana et al. (2018) proposed a method for door detection from 3D colored point clouds data. Their approach could detect open, semi-open, and closed doors from the laser scanned data of an indoor environment by integrating geometry, colors, and other characteristics into the detection analysis. As a result, they were able to detect doors with close to 100% precision and higher than 90% recall. Adán et al. (2018) proposed a method for detecting secondary building components (e.g., MEP components) from laser scanners. They proposed a 6D approach (XYZ+RGB) to recognize small objects such as switches and signs by inferring the objects following a consensus procedure, to create an as-is semantically rich 3D model. Puente et al. (2014) proposed a method for detecting road tunnel luminaires using mobile Light Detection and Ranging (LiDAR) technology. Hamledari et al. (2017) proposed a method to detect components of under-construction indoor partitions using computer vision-based technologies. It was found that the methods for detecting and constructing as-is building components have been studied extensively with practical results. However, reasoning about semantic building concepts (e.g., egress) from as-designed models was under-researched to produce reliable and practical results, as discussed in the information exchange section. The detection of building code concepts is also different from detecting objects, because the building code concepts may involve multiple objects, relationships between them, and their combinations. As a result, new methods need to be developed to help identify these building code concepts, which the existing methods did not cover.

2.4.3 IFC object classification

To address the misuse in IFC entities, a high-performing classification algorithm is needed to label the IFC data with their intended semantic category based on the information extracted from that object. Several attempts have been conducted to develop such algorithms.

From a historical perspective, object classification makes a difference between BIM and 3D computer-aided design (CAD), because BIM carries the semantic information of objects in a building model (Ma et al. 2018).

A few recent works in IFC object classification were found. For example, Koo and Shin (2018) explored the use of a machine learning approach to detect IFC object misclassifications during manual creation of the IFC data. They used SVMs to classify objects into individual classes and tested the SVMs on four classes, which are 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. (2018) 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.

While high performance has been achieved for the machine learning method, it is still not accurate enough to support seamless BIM interoperability. A more systematic and comprehensive feature set may improve the accuracy of the machine learning algorithms. In addition, a more systematic selection of machine learning algorithms needs to be conducted to select the best machine learning algorithm for AEC object classification. For the rule-based algorithms, the efficiency needs to be improved, and the results need to be tested with more data. A comparison of the rule-based algorithm and the machine learning algorithms in the same dataset can provide more evidence in selecting the best approach.

2.5 Machine Learning

2.5.1 Machine learning algorithms

Machine Learning (ML) plays an important role in the success of many modern computing technologies such as artificial intelligence (AI). There are traditional statistical methods such as linear regression and logistic regression, and new methods that require large computational efforts such as convolutional neural networks (CNN) (Kuang and Xu 2018) and generative adversarial networks (GAN) (Seeliger et al. 2018). Machine learning algorithms can be categorized into two types - supervised and unsupervised. Supervised algorithms require ground truth as input, which is the correct labels of the data. Unsupervised algorithms do not have such prerequisite. Based on the purpose of application, machine learning algorithms can also be categorized by tasks, e.g., classification, regression, and clustering. Kotsiantis (2007) reviewed major machine learning algorithms for the classification task: logic-based algorithms, including decision trees and rule-based algorithms; perception-based algorithms, including logistic regression (Grégoire 2014), neural networks (Cartwright 2015, Cartwright, 2008, Angermueller et al. 2016, Zeng et al. 2019, Silver et al. 2016, Rosandich 1997, and Mahanta 2017), and radial basis function networks; statistical learning algorithms, including naive Bayes and Bayesian network (Kotsiantis 2007, Quinlan 1986, and Jensen 1996); instance-based learning, including nearest neighbors and k-nearest neighbors (Veropoulos et al. 1999); and SVM. Kotsiantis (2007) summarized the properties and method of each type of algorithms and provided analysis of each of them. For supervised object classification, multiple existing algorithms provide accurate results for different cases. Some of the most promising ones include logic-based algorithms, perception-based techniques, statistical learning algorithms, instance-based algorithms, and SVM (Kotsiantis 2007). In addition to those ML algorithms, boosting method can increase the overall accuracy by reducing variance (Hastie et al. 2009) and therefore can be used to develop new ML algorithms. For example, Random Forest (Hastie et al. 2009) is a machine learning algorithm that is built on substantial modification of the bagging method which in turn is also known as bootstrap aggression to reduce the variance of an estimating function. The bagging method works well for tree structures. Random forest pushes that further to build a large collection of de-correlated trees and take the mean prediction of the many trees in classification results using the bagging method.

2.5.2 Feature engineering

To fully explore the potential of machine learning algorithms, a systematic feature engineering is needed.

Feature engineering is one of the critical steps to ensure that the machine learning algorithms can generate good models to achieve the desired classification results. In addition, Feature engineering is one of the most time-consuming and challenging tasks in data mining (Zhang et al. 2018). According to the Occam's Razor (Bethel 2009), the fewer features used, the more robust machine learning algorithms can potentially be, so the main task in feature engineering is to select a small set of features that maintain a good performance in the target machine learning task. Koo et al. (2019) used a feature set consisted of certain geometric information and relational information.

2.6 Logic Programming

The formal foundations of logic programming started in the late 1970's and further developed in the early 1980's (Alferes 1996). With declarative nature, logic programming became a candidate for knowledge representation. In addition, the relations with deductive database made logic programming even more suitable for knowledge representation. This approach provided machines with a logic representation of the knowledge and made the reasoning independent of any particular implementation. It is context-free and easy to manipulate. Among the implementations of logic programming, Prolog (Max 2013 and Spivey 1996) is the most widely used logic programming language. It does not require a background in mathematics, logic, or artificial intelligence (AI) to use.

2.7 Automated Compliance Checking of Building Codes

2.7.1 Domain knowledge representations of building codes

To enable ACC, one of the main steps is to convert the building codes written in natural/human language into computing language. In natural language's discussion, syntax refers to the word sequences, semantics refers to the sense and meaning, and pragmatics refers to different interpretations in different contexts (Nawari 2018). To enable machines to process such syntax, semantics, and pragmatics, natural language processing (NLP) (Nadkarni et al. 2011) methods were developed and used. NLP approach has been shown to be promising in converting building codes rules and infrastructure regulations written in plain text into computable representations (Zhang and El-Gohary 2016b; Xu and Cai 2020; Xue and Zhang 2020, 2021). Regarding computable representations of building codes, a lot of research has been conducted in representing the building code requirements in various computable formats. For example, Khemlani (2011) proposed to use predicate logic to represent building codes in FORNAX (i.e., a C++ library for IFC data editing). Ding et al. (2006) proposed to use rule-based language to represent accessibility requirements in building codes. Martins and Monteiro (2013) proposed to use XML-based language to represent building codes of water systems. Tan et al. (2010) proposed to use XML-based language to represent building codes for checking building envelope using the Jboss rule engine. In addition, Jeong and Lee (2009) proposed to use direct hard coding to check building fire resistance requirements automatically. Nawari (2012) proposed to use object-oriented representation for encoding a knowledge domain. Zhang and El-Gohary (2016c) proposed to use first-order logic (FOL) for encoding building codes. FOL consists of objects, relations, and functions. In recent research, domain-specific ontology was also used in constructing a set of semantic and syntactic structures for building codes (Zhang and El-Gohary 2016c). Independent from such computable representations, however, is the need of matching BIMs to related building code concepts.

2.7.2 Automated building code compliance checking (ACC)

The first successful effort for ACC can be traced back to the 1960s when Fenves (1966) designed an if-then system to represent American Institute of Steel Construction (AISC) standard specifications. Later, many research efforts followed and made advancement in ACC (Lopez and

Wright, 1985; Lopez et al., 1989, Garrett and Fenves, 1987). In recent studies (Holzer 2015, Sionov et al. 2015, Volk et al. 2014, Zou et al. 2017, and Sacks 2018), researchers showed great interests in using BIM to support ACC, which is expected to be interoperable among platforms for designers, contractors, clients, vendors, and others. An important problem each and every BIM-based ACC system must address, is how to match building design information from BIMs to building code concepts that are essential building blocks of regulatory rules in building codes. This was reflected in the four-stage rule checking framework that Eastman et al. (2009) summarized which included "(1) rule interpretation and logical structuring of rules for their application; (2) building model preparation, where the necessary information required for checking is prepared; (3) the rule execution phase, which carries out the checking, and (4) the reporting of the checking results." Both the first two stages are intended to prepare input (i.e., building design and building codes, respectively) for the third phase – rule execution. In order for the building code rules to execute over the building design input, there lies the matching between the two inputs. In tackling this problem, recent ACC efforts mainly endeavored at the rule level (i.e., regulatory requirements from building codes). For example, Kasim et al. (2013) proposed a reusable solution for sustainability compliance checking using the requirement-applicabilities-selection-exception (RASE) methodology developed by Hjelseth and Nisbet (2011). Their method can extract sustainability requirements and convert them into compiled rules for use with a rule engine. It has the potential to support dynamic checking during the building design stage. Solihin and Eastman (2015) proposed to classify the rules in building codes into six categories, including: (1) rules for checking the well-formedness of a building model, (2) rules for building regulatory code checking, (3) rules for constructability/other contractor requirements, (4) rules for safety/other rules with corrective actions, (5) rules for warranty approvals, and (6) rules for checking BIM data completeness. Zhang and El-Gohary (2017) proposed a semantic NLP and logic reasoning-based method to support fully ACC. Their prototype achieved 98.7% recall & 87.6% precision in noncompliance detection in Chapter 19 of the International Building Code 2009 which was based on representing each regulatory requirement as a logic rule. Haubler et al. (2021) proposed a rule-based method to implement the ACC of railways, with 12 categories of rules (e.g., component definitions, directional definitions). Their approach was shown to be able to automatically examine 37%-75% of the 943 rules. Xue et al. (2021) developed a semi-automated method for information extraction from tabular contents of building codes and information transformation into computable rules. Their method correctly processed 91.67% of tables when tested on Chapter 10 of the International Building Code 2015.

BIM has a great potential in supporting ACC, especially with the support of IFC standard, which is open and platform-neutral. However, despite the existing efforts and progresses achieved, practical obstacles remain, in information extraction and matching from both BIMs and specific building codes, at the regulatory concept level. For example, BIMs are not expected to contain explicit depictions of “egress”, which is an important concept in building codes.

CHAPTER 3 – A NEW AEC OBJECT DATASET

Significant portions of this chapter can be found in:

“Automated BIM object classification to support BIM interoperability.” In *Proc., Construction Research Congress*, 706-715. DOI: 10.1061/9780784481301.070.

“New automated BIM object classification method to support BIM interoperability.” in *Journal of Computing in Civil Engineering*, 33(5). DOI: 10.1061/(ASCE)CP.1943-5487.0000858.

“Introducing geometric signatures of architecture, engineering, and construction objects and a new BIM dataset.” In *Proc., 2019 ASCE International Conference on Computing in Civil Engineering*, 264-271. DOI: 10.1061/9780784482421.034.

In order to perform a data-driven approach for the experiments, the author built a data set which targets collecting a uniform AEC dataset with broad coverage of types and representations in IFC. The dataset is reusable for reproducing and conducting further research.

3.1 Data Collection: Collect IFC Models from Different Sources to Create a Dataset with a Broad Coverage of Different Types of IFC Entity Usage.

To collect needed data, the author explored existing open BIM repositories, including the “Open IFC Model Repository” (Dimyadi and Henderson 2012) and the NBS National BIM Library (2018), which contain 105 and 6,660 IFC data instances, respectively. They provided good quality models for visualization. However, these data were not tailored for object classification, because they did not have verified object class labels. In contrast, the author developed a new dataset tailored for BIM object classification. The author invited independent annotators to manually label the collected data and discussed among themselves any disagreement. If any disagreement could not be resolved through discussion, the majority vote mechanism was used to decide the label to adopt.

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 author collected data from two 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.

Among the collected data, the author selected the duplex apartment model (Duplex_A) from the first source, which is a model from the buildingSMART official website (WBDG 2021); the Revit (Autodesk 2021) 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, which covers different types of projects. The similarity between all the selected models was that they all contained many beams, columns, footings, slabs, and walls. The selected models contained hundreds of AEC objects on average. AEC objects from the selected models covered many variations of IFC representation used, including *IfcFacetedBrep*, *IfcExtrudedAreaSolid*, *IfcMappingRepresentation*, and *IfcBooleanResult*. For the coverage of object shapes, the data covered a wide range of different shapes with hundreds of unique AEC objects for each type.

3.2 Label the Data with Inter-Annotator Agreement

In order to classify the objects of an IFC model, algorithm needs to be developed to detect the objects and extract all related information. The extraction of IFC objects is achieved using the algorithm of Won et al. (2013), as reproduced by the author, that can extract all building elements from an IFC file and store each element as a separate file for the purpose of labeling. 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). 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 can be identified.

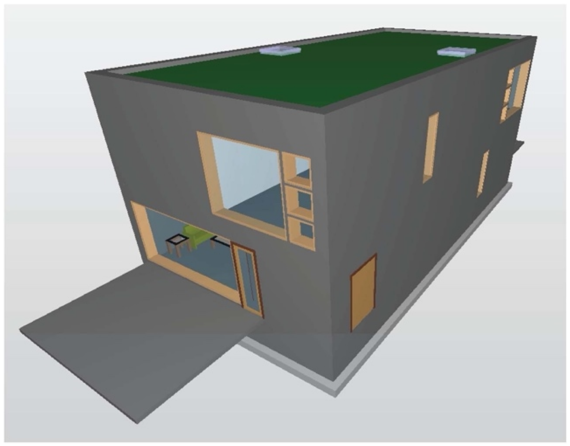


Fig. 4. Visualization of a duplex apartment model.

The author invited three 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 1). For the objects that had different labels by different annotators, the author 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 author picked the majority labels (examples in Table 2). BIM viewer (Fieldwire 2021) was used to visualize the extracted objects and display their properties during the manual labeling. The results are showed in Table 3.

Table 1. Inter-annotator agreements of IFC objects manual labeling before discussion -> after discussion.

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 2. 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 3. Numbers of instances in each object types.

Object Type	Number of Instances
Beam	790
Column	412
Footing	354
Slab	79
Wall	265
Total	*1,900*

3.3 Data Sharing

To facilitate data sharing, the author hosted this open dataset at Purdue University Research Repository (PURR), an initial and a second version of which was described in (Wu and Zhang 2018b, Wu and Zhang 2021b). The new dataset can be used directly by researchers and developers to develop and test object classification algorithms. The dataset contains not only the 1,900 IFC instances of beams, columns, footings, slabs, and walls but also the object class type labels as described below in Chapter 3.2, and selected object features (invariant signatures) as described in Chapter 4.

CHAPTER 4 - INVARIANT SIGNATURES OF AEC OBJECT

Significant portions of this chapter can be found in:

"Constructing invariant signatures for AEC objects to support BIM-based analysis automation through object classification." in *Journal of Computing in Civil Engineering*, submitted.

"Introducing geometric signatures of architecture, engineering, and construction objects and a new BIM dataset." In *Proc., 2019 ASCE International Conference on Computing in Civil Engineering*, 264-271. DOI: 10.1061/9780784482421.034.

To address the gap of lacking intuitive uniform interoperable representations that can be used for different BIM tasks, the author proposes the concept of invariant signatures for AEC objects.

4.1 Invariant Signature Definition

The invariant signatures concept was first conceived by Dr. Jiansong Zhang in his NSF proposal titled "EAGER/Collaborative Research: Science-Based Exploration of Invariant Signatures of Architecture/Engineering/Construction Objects to Enable Interoperability of Building Info Modeling" (NSF 2017). It is inspired by neural signatures and mathematical signatures.

Neural signatures. The structure and functions of human brains are still underexplored. But the way that a human brain detects objects gradually got unraveled, i.e., patterns consisted of features that help people recognize objects (Brandman and Peelen 2017). For example, Johnson and Olshausen (2003) observed this object detection process by a human brain through experiments. In their experiment, two ways to measure event-related potential (ERP), i.e., the electrophysiological response to a stimulus, were used to correlate brain activities with object recognition. Two types of components in an ERP of natural images were discovered: early presentation-locked signal and later recognition-related component, respectively. An early presentation-locked signal indicates low-level feature differences between images. A later recognition-related component covariates with the subsequent reaction time. Their experiment inferred that the second type of neural signatures for image recognition have a substantially later and variable time of onset comparing to the first type, which provides insights into object detection by human brains using neural signatures.

Mathematical signatures. Compared to neural signatures described above, mathematical signatures follow a more concise and rigid formulation. The creation of mathematical signatures followed a rigorous procedure of definition and proof, based on element axioms and complicated deductions. For example, the mathematical signature of a circle includes the following two rules depicting the patterns of features: (1) the set of points that forms the circle must be in the same plane; (2) the boundary (circumference) must be equidistant to a center point (Coolidge 1902). Furthermore, Daniyarova et al. (2012) showed that the entire properties of algebraic universal geometry can be carried over to the case of an arbitrary geometric signature without essential changes.

Other signatures. The idea of signatures is widely used and has a variety of different nature. For example, Stow et al. (2012) proposed frequency distribution signatures for use in the classification of within-object pixels. Nelson and Sokkappa (2008) proposed radiation signatures that were generated using a statistical model to detect nuclear threats. Marat and Ltti (2012) established object signatures for object classification and showed that the amount of context learned had an important effect in object recognition results.

In the same spirit, the author proposed to construct invariant signatures of BIM-based AEC objects, which are "a set of intrinsic properties of the object that distinguish it from others and that do not change with data schema, software implementation, modeling decisions, and/or language/cultural contexts" (Wu et al. 2021). The invariant signatures consist of three sub-types, namely, (1) geometric signatures, (2) locational signatures, and (3) metadata signatures. Geometric signatures capture the shape information such as common shapes as rectangle and cylinder. Locational signatures capture the position information of an object, including the absolute position, relative position, and orientation. Metadata signatures capture representation-level information, especially the representation used in IFC, e.g., the average number of vertices among faces in a boundary representation (Brep).

4.2 Invariant Signature Features

The invariant signatures include both categorical features and numeric features. Categorical features can be transformed into numeric values using discrete numbers or binary representations. For features that an object does not have (i.e., no feature value exists), a default value of "0" or "false" was assigned. In addition, metadata signatures may have certain information overlap with geometric and locational signatures. For example, if an object has nonzero values for the I-shape signatures then the object should also have a "true" value for the extruded area solid signature. Table 4 shows all the developed invariant signatures with their value types, signature types, and meanings. Fig. 5 and Fig. 6 shows the visualization of a few sample properties for I-shape and ring shape.

Table 4. Developed object invariant signatures.

Invariant Signature Name	Value Type	Signature Type	Description
Rec_L	Numerical	Geometric	Length of a rectangular shape
Rec_W	Numerical	Geometric	Width of a rectangular shape
Rec_H	Numerical	Geometric	Height of a rectangular shape
Cir_R	Numerical	Geometric	Radius of a cylinder shape
Cir_H	Numerical	Geometric	Height of a cylinder shape
R_R	Numerical	Geometric	Radius of a ring shape
R_H	Numerical	Geometric	Height of a ring shape
R_T	Numerical	Geometric	Thickness of a ring shape
I_W	Numerical	Geometric	Overall width of an I-shape
I_H	Numerical	Geometric	Height of an I-shape (extruded depth)
I_D	Numerical	Geometric	Overall depth of an I-shape
I_R	Numerical	Geometric	Fillet radius of an I-shape
I_WT	Numerical	Geometric	Web thickness of an I-shape
I_FT	Numerical	Geometric	Flange thickness of an I-shape
X1	Numerical	Locational	Vector x for placement
X2	Numerical	Locational	
X3	Numerical	Locational	
Z1	Numerical	Locational	Vector z for placement
Z2	Numerical	Locational	
Z3	Numerical	Locational	
O1	Numerical	Locational	Center Cartesian point of the object
O2	Numerical	Locational	
O3	Numerical	Locational	
mHigh	Numerical	Locational	Highest elevation of the original model

Table 4. Continued

Invariant Signature Name	Value Type	Signature Type	Description
mLow	Numerical	Locational	Lowest elevation of the original model
mRatio	Numerical	Locational	mRatio = (mHigh – (mHigh + mLow) / 2) / (mHigh - mLow)
Length	Numerical	Geometric	Length of the bounding box
Width	Numerical	Geometric	Width of the bounding box
Height	Numerical	Geometric	Height of the bounding box
Volume	Numerical	Geometric	Volume of the bounding box
Items	Integer	Metadata	Number of items
Faces	Integer	Metadata	Number of faces of Brep representation
F3	Integer	Metadata	Number of faces with 3 edges of Brep representation
F4	Integer	Metadata	Number of faces with 4 edges of Brep representation
F7	Integer	Metadata	Number of faces with 7 edges of Brep representation
AveVerti	Numerical	Metadata	Average number of edges of all faces of Brep representations
Xmax	Numerical	Metadata	Max value in x direction
Xmin	Numerical	Metadata	Min value in x direction
Ymax	Numerical	Metadata	Max value in y direction
Ymin	Numerical	Metadata	Min value in y direction
Zmax	Numerical	Metadata	Max value in z direction
Zmin	Numerical	Metadata	Min value in z direction
Brep	Nominal (Binary)	Metadata	If Brep representation is used
Extruded	Nominal (Binary)	Metadata	If swept solid representation is used
Clipping	Nominal (Binary)	Metadata	If clipping representation is used
CSG	Nominal (Binary)	Metadata	If CSG representation is used
SurfaceModel	Nominal (Binary)	Metadata	If SurfaceModel representation is used
ExtrudX	Numerical	Geometric	
ExtrudY	Numerical	Geometric	Extruded direction
ExtrudZ	Numerical	Geometric	
Rec	Nominal (Binary)	Metadata	If a rectangular shape is used
Cir	Nominal (Binary)	Metadata	If a cylinder shape is used
Ring	Nominal (Binary)	Metadata	If a ring-shape is used
I	Nominal (Binary)	Metadata	If an I-shape is used
Type	Nominal (Quinary)	Ground Truth	Labeled type

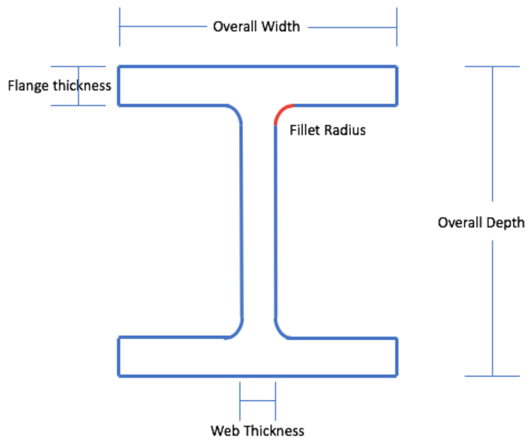


Fig. 5. Cross-sectional properties of an I-shape.

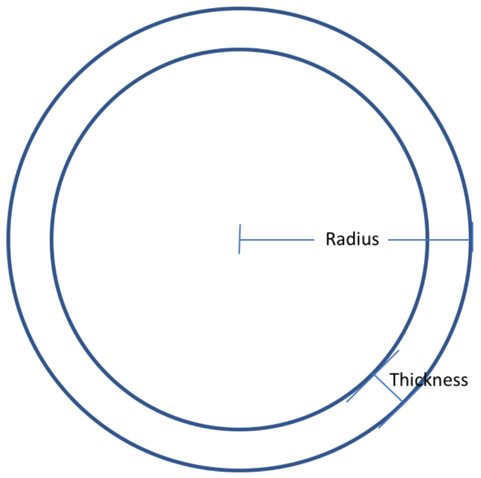


Fig. 6. Cross-sectional properties of a ring shape.

The signature set contains built-in feature values, such as *Rec_L*, *Rec_W*, *Rec_H*, which represent the length, width, and height of a 3D rectangular shape, respectively. For regular shapes, rectangular, cylindrical, and ring shape features were included. For irregular shapes, the author defined two sets of signatures, one for extruded area solid and one for faced boundary representation (Brep). The extruded area solid signature for irregular shapes included a set of 3 decimal numbers to represent the length, width, and thickness dimensions of the bounding box, respectively. For Brep, in addition to the dimensions of the bounding box, the signatures that indicate the number of faces were also included. Table 5 shows 16 examples of the invariant signature values of independent objects.

Table 5. Sample invariant signature values by instances.

Instance	Model	*Rec_L*	*Rec_W*	*Rec_H*	*Cir_R*	*AveVerti*	*Type*
IfcFooting1	Duplex_A	18.283	0.9	0.3	0	0	Footing
IfcFooting2	Duplex_A	8.383	0.9	0.3	0	0	Footing
IfcFooting3	Duplex_A	17.383	0.9	0.3	0	0	Footing
IfcWall1	Duplex_A	16.966	0.417	1.25	0	0	Wall
IfcWall2	Duplex_A	4.2005	0.435	1.25	0	0	Wall
IfcSlab1	Rac_Basic	0	0	0	150	0	Column
IfcSlab2	Rac_Basic	0	0	0	150	0	Column
IfcSlab3	Rac_Basic	0	0	0	150	0	Column
IfcBeam174	Rst_Basic	0	0	0	15	0	Beam
IfcBeam114	Rst_Basic	0	0	0	0	3.519503546	Beam
IfcBeam115	Rst_Basic	0	0	0	0	3.52228164	Beam
IfcBeam116	Rst_Basic	0	0	0	0	3.52228164	Beam
IfcBeam117	Rst_Basic	0	0	0	0	3.519503546	Beam
IfcWall136	Tech_School	0	0	0	0	3.007682815	Wall
IfcWall137	Tech_School	0	0	0	0	3.009665124	Wall
IfcWall138	Tech_School	0	0	0	0	3.028989751	Wall

The proposed invariant signatures are expected to uniquely identify AEC objects. To allow this identification, the invariant signatures shall describe all the major information embedded in each object. This can be reflected by the statistical relations of each invariant signature with AEC object types. For one type of AEC object, the invariant signature values fall into a certain range. For example, the height of a slab object usually does not exceed 1 ft. However, the height of a wall object usually does not fall below 1 ft. This is reflected in the distribution of each invariant signature value. Figs. 7 to 9 show three plots of object instances' distributions across three different invariant signature features, respectively. Fig. 7 shows the distribution of the locational signature *O3* (elevation of an object) on different object types. It shows that footings have smaller values of elevation. Fig. 8 shows the distribution of the geometric signature *Length* (horizontal dimension) on different object types. It shows that columns have smaller values of length, comparing to other types. Fig. 9 shows the distribution of the geometric signature *Cir_R* on different object types. It shows that circle is not used in walls or slabs.

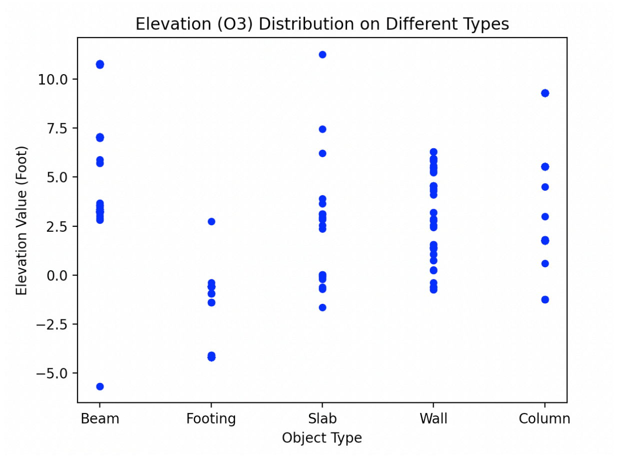


Fig. 7. Distribution of the locational signature *O3* (elevation) on different object types.

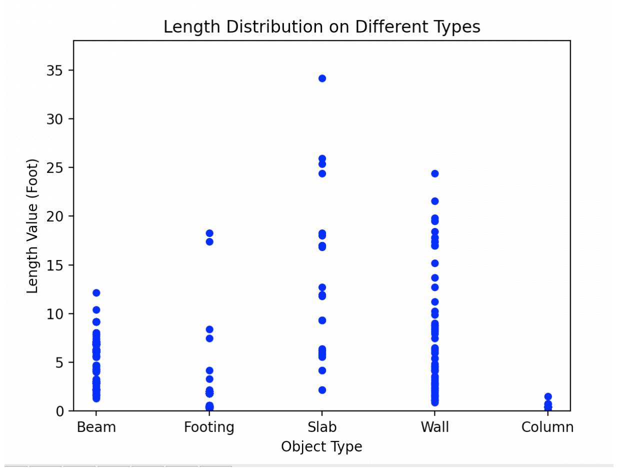


Fig. 8. Distribution of the geometric signature *Length* on different object types.

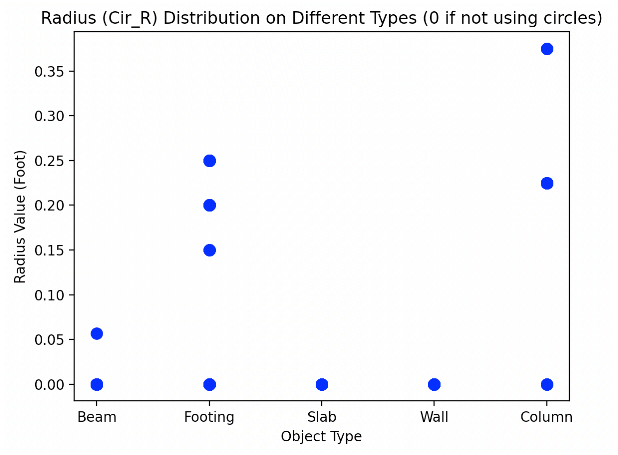


Fig. 9. Distribution of the geometric signature *Cir_R* on different object types.

CHAPTER 5 - INVARIANT SIGNATURE – BASED AEC OBJECT CLASSIFICATION

Significant portions of this chapter can be found in:

"Automated BIM object classification to support BIM interoperability." In *Proc.*, *Construction Research Congress*, 706-715. DOI: 10.1061/9780784481301.070.

"New automated BIM object classification method to support BIM interoperability." in *Journal of Computing in Civil Engineering*, 33(5). DOI: 10.1061/(ASCE)CP.1943-5487.0000858.

"Constructing invariant signatures for AEC objects to support BIM-based analysis automation through object classification." in *Journal of Computing in Civil Engineering*, submitted.

With the theoretical concept and its potential implementation of invariant signatures, it is essential to test the robustness of the invariant signatures in solving practical BIM problems. New automation methods are proposed for different BIM tasks with the adoption of invariant signatures. In this chapter, BIM object classification is investigated, to support the automated detection of IFC entity misuses.

5.1 Rule-Based Method for AEC Object Classification

The author proposes a new seven-step iterative method to classify BIM objects in IFC models (Fig. 10): 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. To help illustrate the method, some implementation examples are used in this explanation.

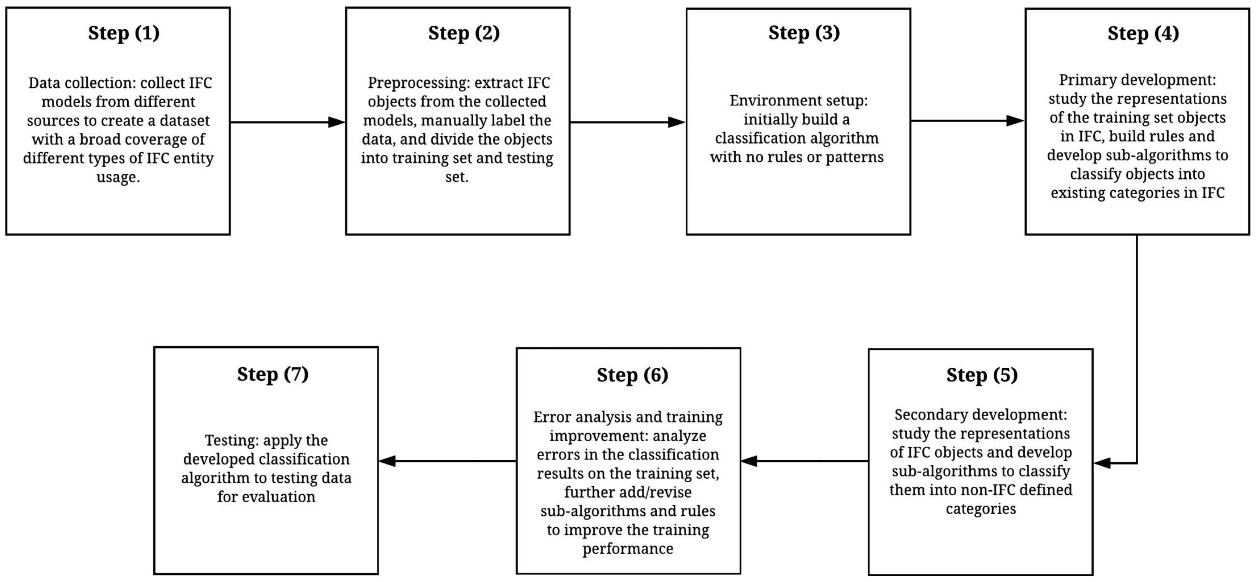

Step (1)
Data collection: collect IFC models from different sources to create a dataset with a broad coverage of different types of IFC entity usage.
Step (2)
Preprocessing: extract IFC objects from the collected models, manually label the data, and divide the objects into training set and testing set.
Step (3)
Environment setup: initially build a classification algorithm with no rules or patterns
Step (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
Step (5)
Secondary development: study the representations of IFC objects and develop sub-algorithms to classify them into non-IFC defined categories
Step (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
Step (7)
Testing: apply the developed classification algorithm to testing data for evaluation

Fig. 10. Proposed 7-step method for automated IFC-based BIM object classification.

5.1.1 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 consisting of models collected from different sources is expected to have a broader coverage of different types of representations and uses of IFC entities compared to models collected from a single source.

5.1.2 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 author, 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. 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 author 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% - 30% (Kemal and Salih 2007). However, the author's rule-based learning has more rationality (i.e., based on geometric theorems) built into the training process and therefore requires less training data compared to statistical learning, in spite of its dependency on the variety of data representations and their distributions. To study such a learning effect, the author proposes the use of a learning curve measure that will be described in detail in the experiment section (Section 5.2).

5.1.3 Environment setup

The author uses 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 because 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 consisting of features that could uniquely recognize a category. These features are inherent properties of the

AEC objects such as number of subcomponents, number of faces, cross-sectional profile, extrusion direction, dimensional ratio, number of straight lines and curves, line connection angle, length, and turn direction. Extraction of these features is achieved using the author's developed object analysis algorithms similar to the object extraction algorithms. At this step, there are no sub-algorithms or rules.

5.1.4 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 main 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. 11): (a) input reading: get an object instance from the training data; (b) sub-algorithm 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 Section 5.1.2; (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; and (f) recursion: get the next object instance from the training data that was not classified and repeat the process until all object instances in the training data are classified.

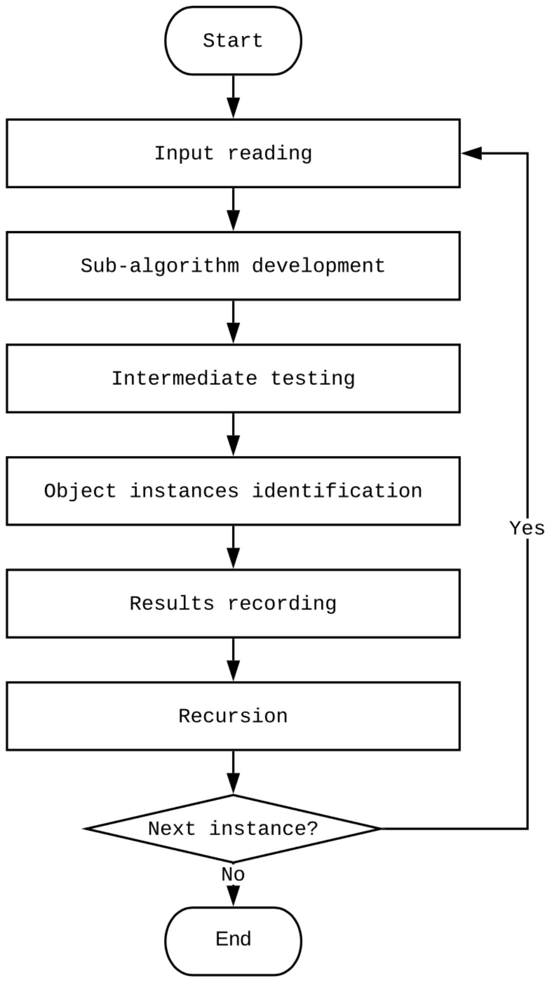


Fig. 11. Primary development flowchart.

5.1.5 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 (Section 5.4.1 Primary Development) will be used. Each developed sub-algorithm will be used to identify one specific object type such as I-beam, C-beam, and rectangular beam.

In addition to the five categories, the author expanded the classification on beams to classify beams into subtypes, including U-Beam, I-Beam, C-Beam, T-Beam, etc. (Chennu 2017; buildingSMART 2018c). The author developed an algorithm for each of the corresponding built-in shape (e.g., I-Shape), swept solid with *IfcArbitraryClosedProfileDef*, and "Brep" representations, respectively.

5.1.6 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 Sections 5.1.4 and 5.1.5 (Primary Development and Secondary Development), 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. 12): (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 are tried.

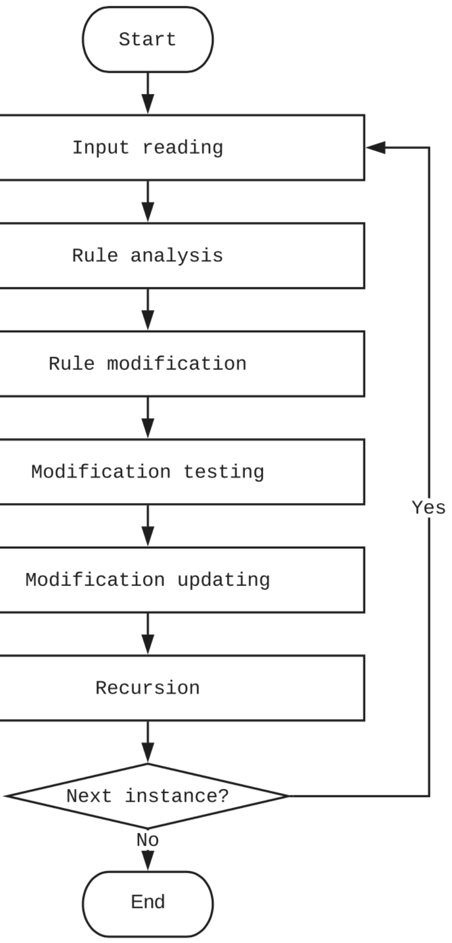


Fig. 12. Error analysis and training improvement flowchart.

5.1.7 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 author adapted the measurements of recall and precision from the 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.

5.2 Rule-Based Method Experiment

5.2.1 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 author used the data from the new proposed dataset described in Chapter 3. In addition, the author also collected models (some special beams) from the National BIM library of UK (NBS of UK 2014), for secondary development. The special beam model objects were collected so that the author could test the classification of sub-types for the proposed method.

5.2.2 Preprocessing: extract IFC objects from the collected models, manually label the data, and divide the objects into training set and testing set

With the proposed dataset in Chapter 3, this step was simplified because extracting and labeling was already finished.

Using the data-dividing Java program that the author wrote, the author collected 1,330 objects into the training set and 575 objects into the testing set. The total is 1,905 objects, which is five more than the 1,900 collected in Chapter 3 with the addition of five extra beams to support secondary development. The collected data was not exhaustive but sufficient for testing the author's proposed method (Beleites et al. 2013), i.e., with more than a hundred instances for each type. 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.

5.2.3 Environment setup: initially build a classification algorithm with no rules or patterns.

The author 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.

5.2.4 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 Section 5.1.4, the author 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 author recorded the number of correctly classified instances after each stage of development and plotted them as a learning curve. Table 6 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. 13 shows the plot of the learning curve. Some development details are described below.

Table 6. Learning curve parameters.

Stage Number	Stage/Content of Study	Number of Objects Classified
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	1,330
Stage 13	Locational Information	1,330
Total	All	1,330

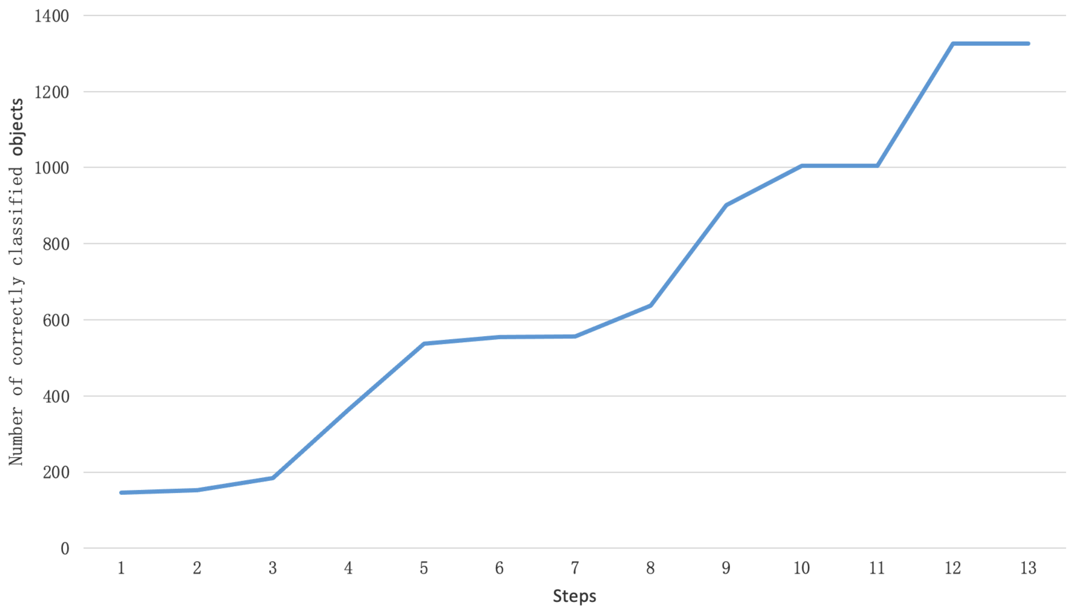


Fig. 13. Plot of the learning curve for all objects in the training data.

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 1,330 (77.82%) 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 (Hébert 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” representation 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 author 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 author used 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 author 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 author combined the extruded direction and placement information using Equation (1) to compute the final extruded direction of the object.
Final extruded direction:

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

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. 14 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 follows “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 vs. circle) and ratios between the three dimensions are used. For example, a circular cross-sectional 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.

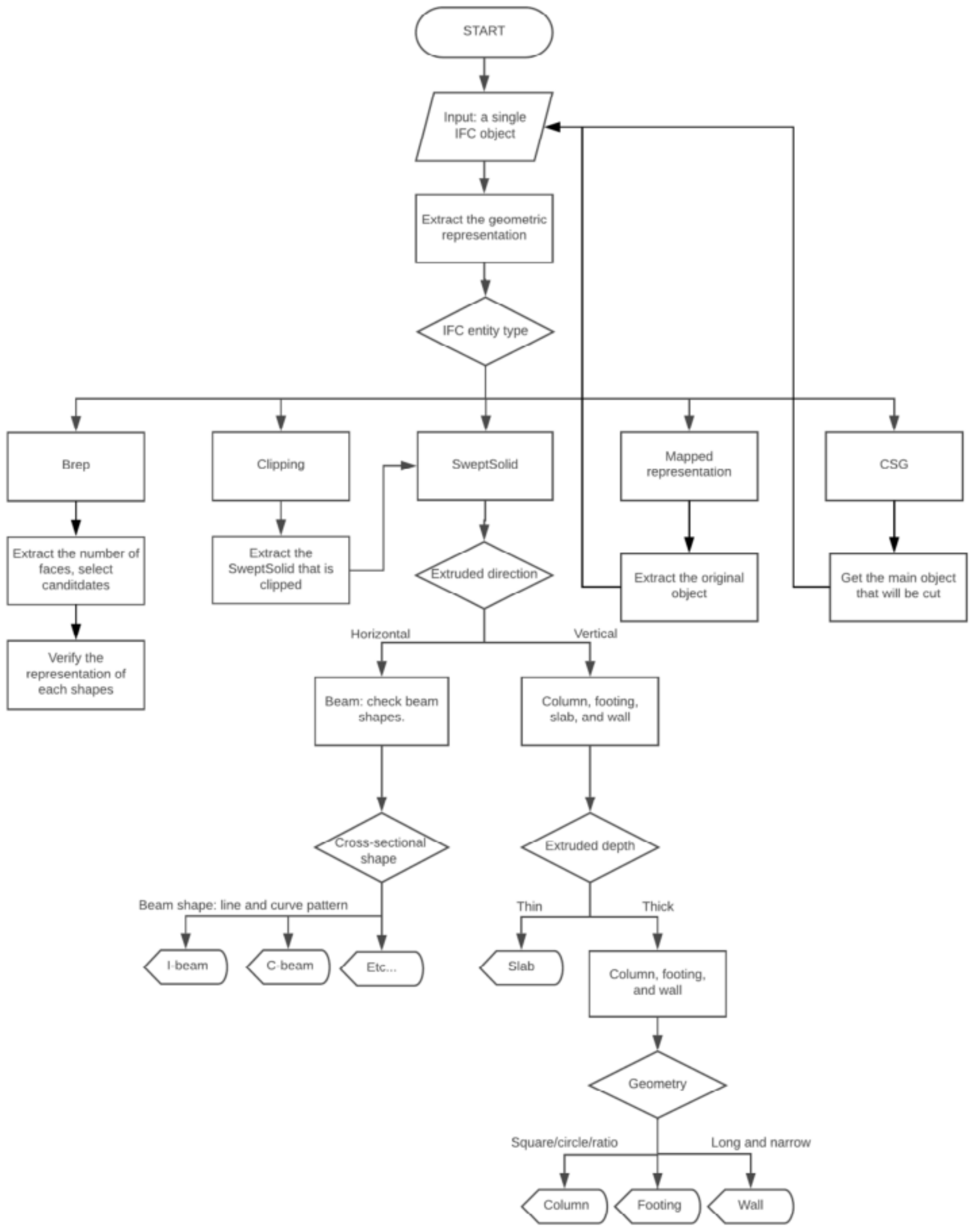

START
Input: a single IFC object
Extract the geometric representation
IFC entity type
Brep
Clipping
SweptSolid
Mapped representation
CSG
Extract the number of faces, select canditdates
Verify the representation of each shapes
Extract the SweptSolid that is clipped
Extruded direction
Extract the original object
Get the main object that will be cut
Horizontal
Vertical
Beam: check beam shapes.
Column, footing, slab, and wall
Cross-sectional shape
Extruded depth
Beam shape: line and curve pattern
Thin
Thick
I-beam
C-beam
Etc...
Slab
Column, footing, and wall
Geometry
Square/circle/ratio
Long and narrow
Column
Footing
Wall

Fig. 14. Algorithm flowchart for rule-based object classification.

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 Section 5.2.5.

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 modifications 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.

In the last step, the author found that 167 footing piers were similar to columns and incorrectly classified into columns, as shown in Fig. 15. This is due to the fact the previous development relied solely on geometric information in such a case. In other words, by checking shape information here, the footing piers cannot be differentiated from columns. The author improved the algorithm to add the consideration of the relative location of an object to other objects.

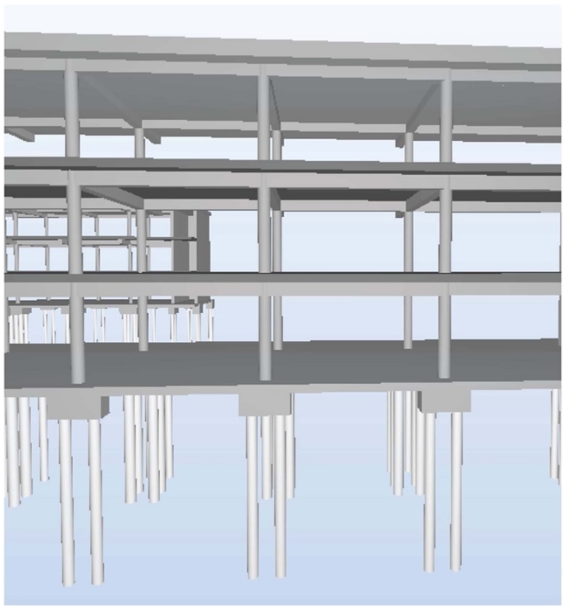


Fig. 15. Visualization of footing piers and columns with the same geometric information.

As a result of the development, the author 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 misuses. Instead, it directly searches for the geometric representation information of the object from the invariant signatures and uses the information for the classification. If corresponding information was not found in the invariant signatures, additional extraction algorithms were developed.

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

In this step, the author 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 author focused 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 author 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/delineated 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. 16.

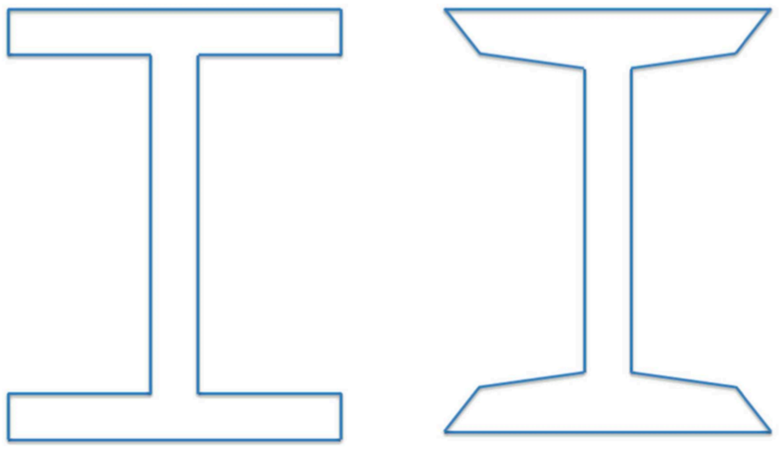


Fig. 16. Types of I-beam: (1) W-section; and (b) S-section.

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 author developed a sub-algorithm that counts the number of boundary lines and curves and checks the linkages between them. The author calls 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. 17, 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 author developed the following four-step method (Fig. 18): (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.

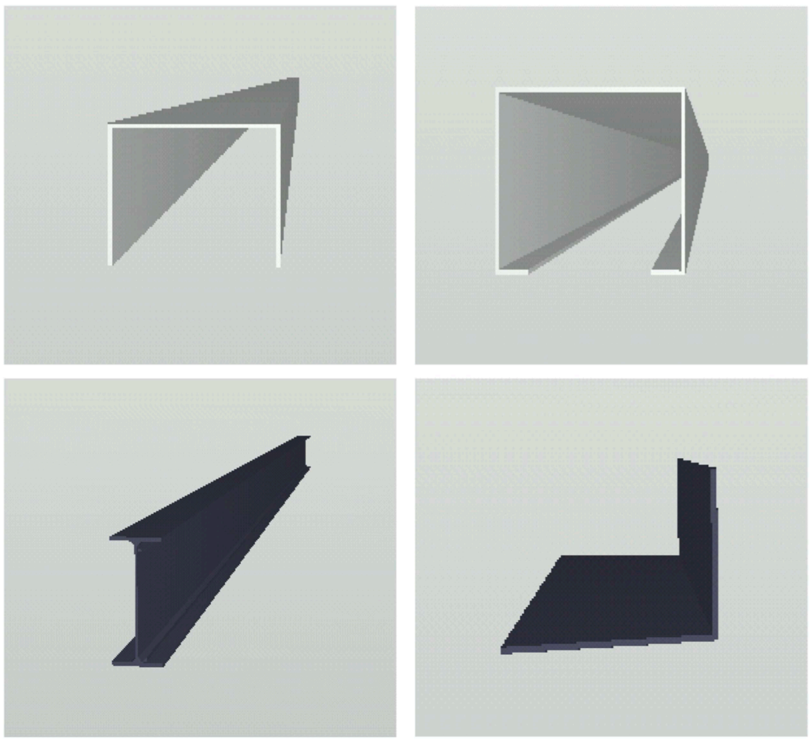


Fig. 17. U-beam, C-beam, I-beam, and L-beam.

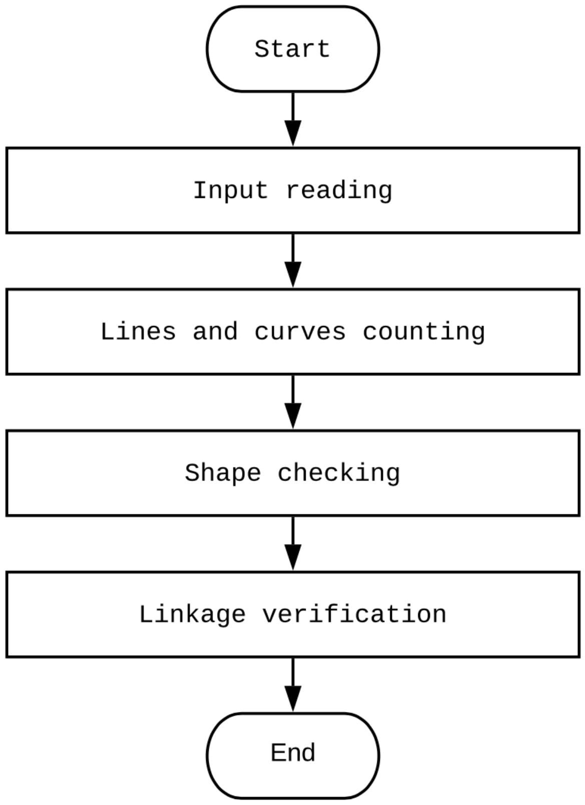


Fig. 18. Beam type classification method.

Among all the beams in the training data, the author studied four shapes that had built-in 2D profiles in IFC. Examples of these shapes are shown in the beams in Fig. 17. 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 7 shows the calculated possible counts of lines and curves of Rectangular Beam, U-Beam, C-Beam, I-Beam, and L-Beam, respectively.

Table 7. 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

Based on the information in Table 7, the author summarized possible beam types of different geometric patterns in terms of the number of lines and curves in their geometric representations (Table 8).

Table 8. 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

According to Table 8, 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 author 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. 19. 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. 19 should not be classified as an I-Beam while the shape in the left part of Fig. 19 should. Such distinction is made at the linkage checking step by analyzing the turn directions of the lines.

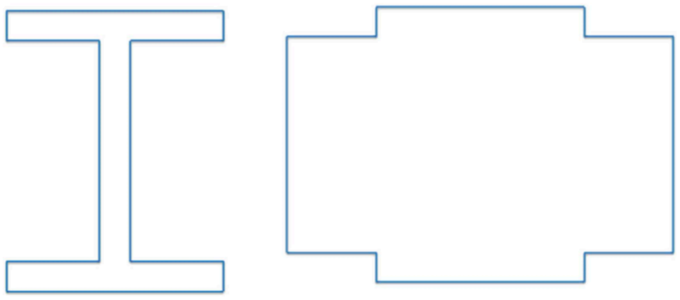


Fig. 19. Shapes with 12 lines but different line connection types.

For Brep, there are many ways to represent a beam, the author 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 author 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. 20 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/chord) and many web members across the bridge, with each major beam consisted of two L-beams as shown in Fig. 20. The author 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 author classified this type of "Beam" into a truss category. As shown in Fig. 21, 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 author developed sub-algorithms for each unique type of beams. Examples of such shapes of beams are shown in Fig. 22.

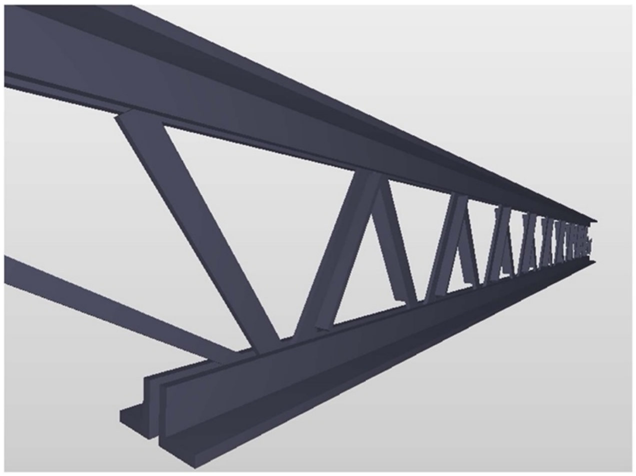


Fig. 20. Visualization of a truss.

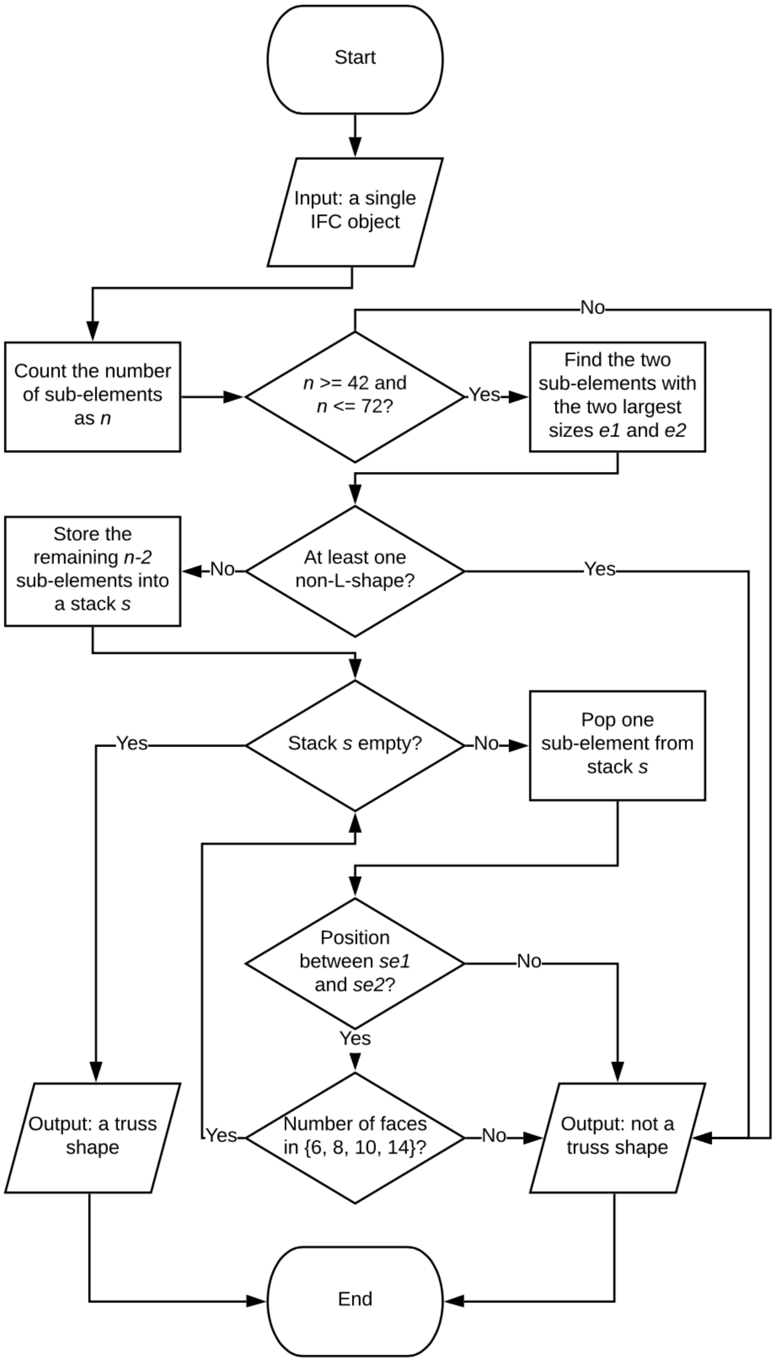

Start
Input: a single IFC object
Count the number of sub-elements as *n*
n >= 42 and *n* <= 72?
Yes
No
Find the two sub-elements with the two largest sizes *e1* and *e2*
At least one non-L-shape?
No
Yes
Store the remaining *n-2* sub-elements into a stack *s*
Stack *s* empty?
Yes
No
Pop one sub-element from stack *s*
Position between *se1* and *se2*?
No
Yes
Number of faces in {6, 8, 10, 14}?
Yes
No
Output: a truss shape
Output: not a truss shape
End

Fig. 21. Developed sub-algorithm for recognizing a truss.

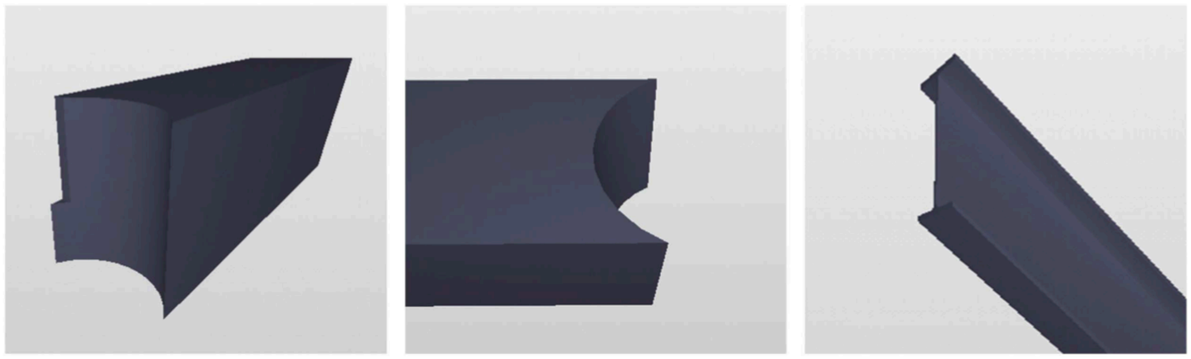


Fig. 22. Other shapes of beams.

5.2.6 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 9).

Table 9. AEC 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*)	Precisio n (*c*/*b*)
Beam	561	561	561	100%	100%
Column	285	285	285	100%	100%
Footing	249	249	249	100%	100%
Slab	52	52	52	100%	100%
Wall	183	183	183	100%	100%
Total	*1,330*	*1,330*	*1,330*	100%	100%

In all the categories of beam, column, footing, slab, and wall, 100% recall and precision were achieved.

The classification results on detailed beam sub-types also achieved 100% recall and precision in all categories, as shown in Table 10.

Table 10. Beam classification results of training data.

Beam Sub-Type	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%*

5.2.7 Testing: apply the developed classification algorithm to testing data for evaluation.

To test the expected performance of the algorithm, the author tested the algorithm on testing data. The results are listed in Table 11 and Table 12.

Table 11. AEC 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	127	127	100%	100%
Footing	105	105	105	100%	100%
Slab	27	25	25	92.59%	100%
Wall	82	79	79	96.34%	100%
Total	*575*	*570*	*570*	*99.13%*	100%

Table 12. Beam 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*)	Precisio n (*c*/*b*)
Rectangula r 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%
Rectangula r 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% 100% 100%	100% 100% 100%
Skewed I-Beam	2	2	2	100%	100%
Total	*234*	*234*	*234*	*100%*	*100%*

5.3 Rule-Based Method Result Analysis and Discussion

The developed algorithm worked well in most cases on the testing data with a higher than 90% precision and recall. For the errors in testing data, the author 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-sectional profile, extrusion direction, dimensional ratio, number of straight lines and curves, line connection angle, length, turn direction, and locations. Using these features, the algorithm can correctly classify beams, columns, footing, slabs, and walls, where the errors come from 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 author's proposed method can result in an accurate and reliable classification method.

A comparison between the proposed method and the methods by Ma et al. (2018) and Sacks et al. (2017) was conducted (Table 13). While all methods work for BIM and could achieve 100% precision and recall, their computational complexities differ. The method by Ma et al. (2018) 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 dissertation 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. (2018) and Sacks et al. (2017).

Table 13. 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. 2018	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.

5.4 Machine Learning Method for AEC Object Classification

For the task of BIM object classification, in this dissertation, the author explores the potential of combining invariant signature-based features and a machine learning approach. In addition to the selection of machine learning algorithms, the selection of features also plays an important role in the performance of machine learning models. With a reported strong performance that was higher than 90%, the feature set used by Koo et al. (2019) seems sufficient. However, exploring a broader range of features could potentially further improve the classification performance, which is crucial in BIM-based automation applications.

To gain more insights into the invariant signatures and to achieve high accuracy in AEC object classification, the author proposes to use invariant signatures and feed them into machine learning algorithms to classify AEC objects. By nature, invariant signatures are expected to suit well the task of AEC object classification because they capture the geometric essence.

Given that the data points in the dataset are at the scale of thousands, the author proposes to use traditional machine learning algorithms instead of deep neural networks, where the latter usually require more data. For algorithm selection, different types of algorithms need to be tested in the training set, and the best-performing algorithm is selected to test on the testing dataset.

5.5 Machine Learning Method Experiment

In the experiment, the author randomly split the 1,900 objects into training dataset and testing dataset following a 7:3 ratio. As a result, 1,330 objects were used as training/development data, and 570 objects were used as testing data. During the development phase, only training data set was used.

With the proposed invariant signatures, five types of machine learning algorithms were tested using Waikato Environment for Knowledge Analysis (Weka), which is an open machine learning platform developed by the University of Waikato (Witten et al. 2016). Ten-fold cross-validation was used to avoid overfitting.

5.5.1. Perceptron-based techniques: neural networks

A single layer artificial neural network would have the same structure with the linear regression model, so the author chose to use neural networks with different layers (not limited to one layer) for perceptron-based techniques to differentiate from linear regression. After parameter tuning, the author selected the best performing configuration to compare with other machine learning algorithms. Table 14 shows the training results of different configurations. Fig. 23 shows a visualization of the accuracies of these results. Table 15 shows the classification results details of the best configuration.

Table 14. Neural network accuracy using different layers and different number of nodes.

Layers\Nodes per Layer	**10**	**20**	**30**	**35**	**40**	**45**	**50**
1	98.05%	98.20%	98.12%	97.89%	98.20%	98.02%	97.97%
2	97.74%	98.05%	98.12%	98.05%	98.35%	98.57%	98.20%
3	96.91%	97.97%	97.67%	97.52%	97.97%	97.29%	97.00%

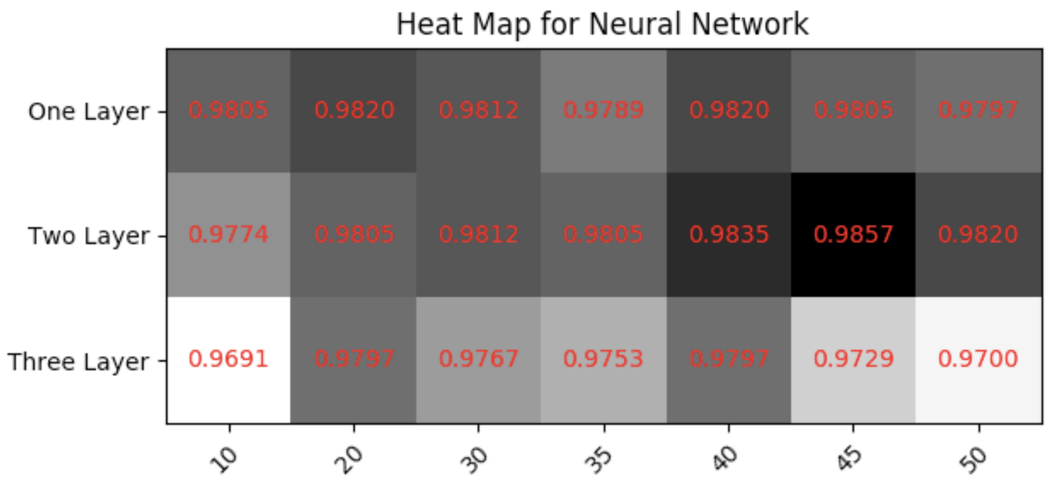


Fig. 23. Visualization of neural network training accuracies on different configurations.

Table 15. Classification results of best configuration of neural networks.

Category	**Number of objects**	**Number of correctly classified objects**	**Number of objects classified into**	**Recall**	**Precision**	**F1**	**ROC**
Beam	549	548	551	99.8%	99.5%	99.6%	100.0%
Column	286	284	286	99.3%	99.3%	99.3%	100.0%
Footing	250	249	255	99.6%	97.6%	98.3%	99.8%
Slab	53	45	49	84.9%	91.8%	88.2%	99.0%
Wall	192	185	189	96.4%	97.9%	97.1%	99.6%
Total	*1,330*	*1,310*	*1,330*	*98.6%*	*98.6%*	*98.6%*	*99.9%*

5.5.2. Logic-based: decision table

A decision table is a graphical implementation of decision trees. The author used best-first search (BFS) and greedy stepwise search. For BFS, the author implemented forward, backward and bidirectional search methods (Table 16). Fig. 24 shows a visualization of the detailed training results. Table 17 shows the details of the classification results of the best configuration.

Table 16. Decision table accuracy using different search direction and depth.

Forward Depth	Accuracy	Backward Depth	Accuracy	Bidirectional Depth	Accuracy
1	97.44%	1	97.74%	1	97.44%
2	97.44%	2	97.74%	2	97.44%
3	97.44%	3	97.74%	3	97.59%
4	97.44%	4	97.74%	4	97.59%
5	97.44%	5	97.74%	5	97.59%
10	97.44%	10	97.74%	10	97.59%

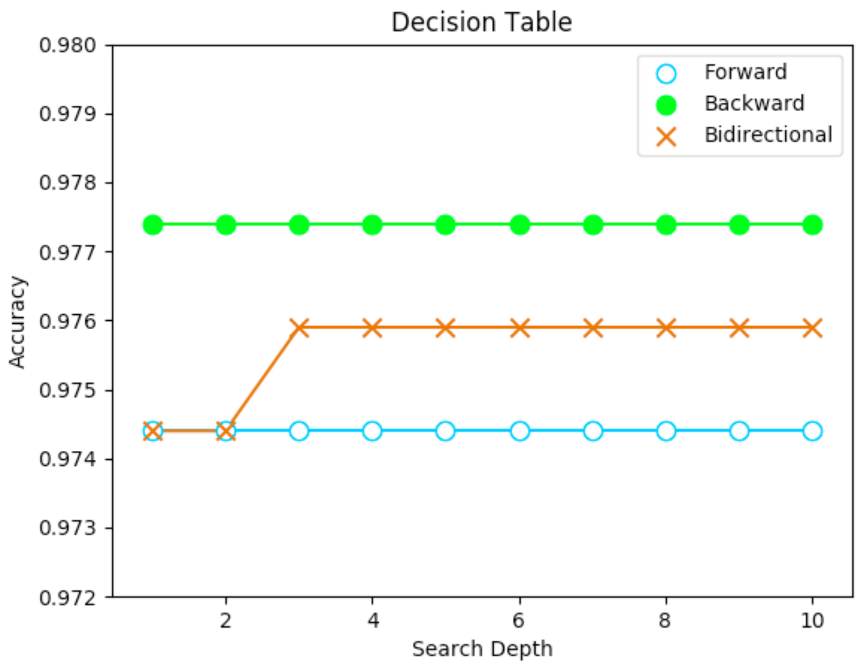


Fig. 24. Visualization of training results of decision table on different configurations.

Table 17. Classification results of the best configuration of decision table.

Category	**Number of objects**	**Number of correctly classified objects**	**Number of correctly classified objects**	**Recall**	**Precision**	**F1**	**ROC**
Beam	549	544	550	99.1%	98.9%	99.0%	100.0%
Column	286	277	280	96.9%	98.9%	98.2%	99.3%
Footing	250	248	260	99.2%	95.4%	97.3%	99.7%
Slab	53	47	48	88.7%	97.9%	93.1%	98.4%
Wall	192	184	192	95.8%	95.8%	95.8%	99.3%
Total	*1,330*	*1,300*	*1,330*	*97.7%*	*97.7%*	*97.7%*	*99.6%*

5.5.3. Statistical machine learning: Bayesian network

Naïve Bayes assumes that the features are independent from each other. However, some of the features may be highly correlated. To address that, the author implemented a Bayesian network instead. Bayesian network used probability graphical model, which was based on the conditional dependencies of the parent nodes on each node in the network. The classification results depend on the number of parent nodes implemented. As a result, the author used different numbers of parent nodes to train the best model. Table 18 shows the training results of different configurations, the visualization of which is shown in Fig. 25. Table 19 shows the details of the classification results of the best configuration.

Table 18. Accuracy vs. number of parents of Bayesian network.

No. of Parents	**Accuracy**
1	96.62%
2	98.80%
3	98.80%
4	98.80%
5	98.80%
6 - 10	98.80%

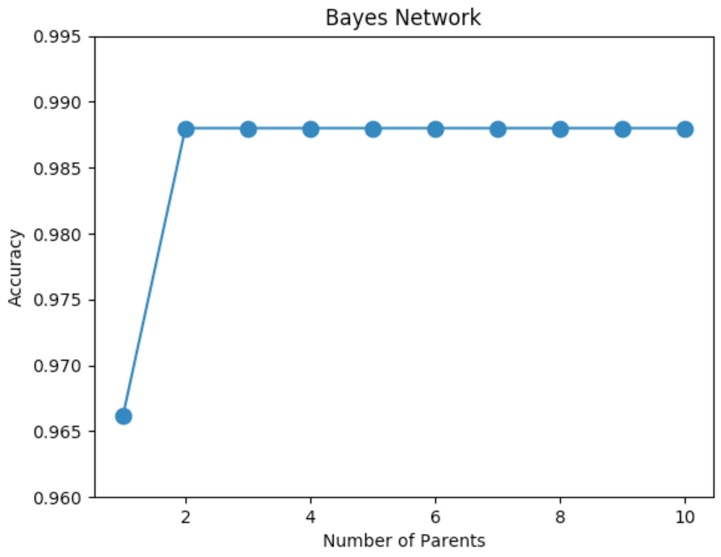


Fig. 25. Visualization of training results of Bayesian network on different configurations.

Table 19. Classification results of the best configuration of Bayesian network.

Category	**Number of objects**	**Number of correctly classified objects**	**Number of objects classified into**	**Recall**	**Precision**	**F1**	**ROC**
Beam	549	547	549	99.6%	99.6%	99.6%	100.0%
Column	286	285	285	99.7%	100.0%	99.8%	100.0%
Footing	250	247	241	98.8%	98.4%	98.6%	100.0%
Slab	53	44	49	83.0%	89.8%	87.3%	99.8%
Wall	192	181	186	99.5%	97.4%	98.5%	100.0%
Total	*1,330*	*1,300*	*1,330*	*98.8%*	*98.8%*	*98.8%*	*100.0%*

5.5.4. Support vector machines (SVM)

SVM uses hyperplanes to separate data into different classes. The classification results depend on a regularization term, which is defined by a soft margin constant. The author experimented with different soft margin constants to find the best performance. Table 20 shows the accuracy of using different soft margin constants at different range scales. A visualization of the accuracy is shown in Fig. 26. Table 21 shows the details of the classification results of the best configuration.

Table 20. Accuracy on different regulation terms C for SVM.

C	Accuracy	C	Accuracy	C	Accuracy	C	Accuracy
1	97.44%	9	98.50%	10	98.57%	14.2	98.65%
5	97.97%	10	98.57%	14	98.65%	14.25	98.72%
10	98.57%	20	98.72%	15	98.72%	14.5	98.72%
50	98.57%	30	98.57%	16	98.72%	14.75	98.72%
100	98.57%	40	98.57%	17	98.65%	15	98.72%
1000	98.20%	50	98.57%	18	98.65%	15.25	98.72%
10000	98.05%	60	98.50%	19	98.65%	15.5	98.72%
		80	98.50%	20	98.72%	15.75	98.72%
		100	98.57%	25	98.65%	16	98.72%
		105	98.50			16.25	98.65%

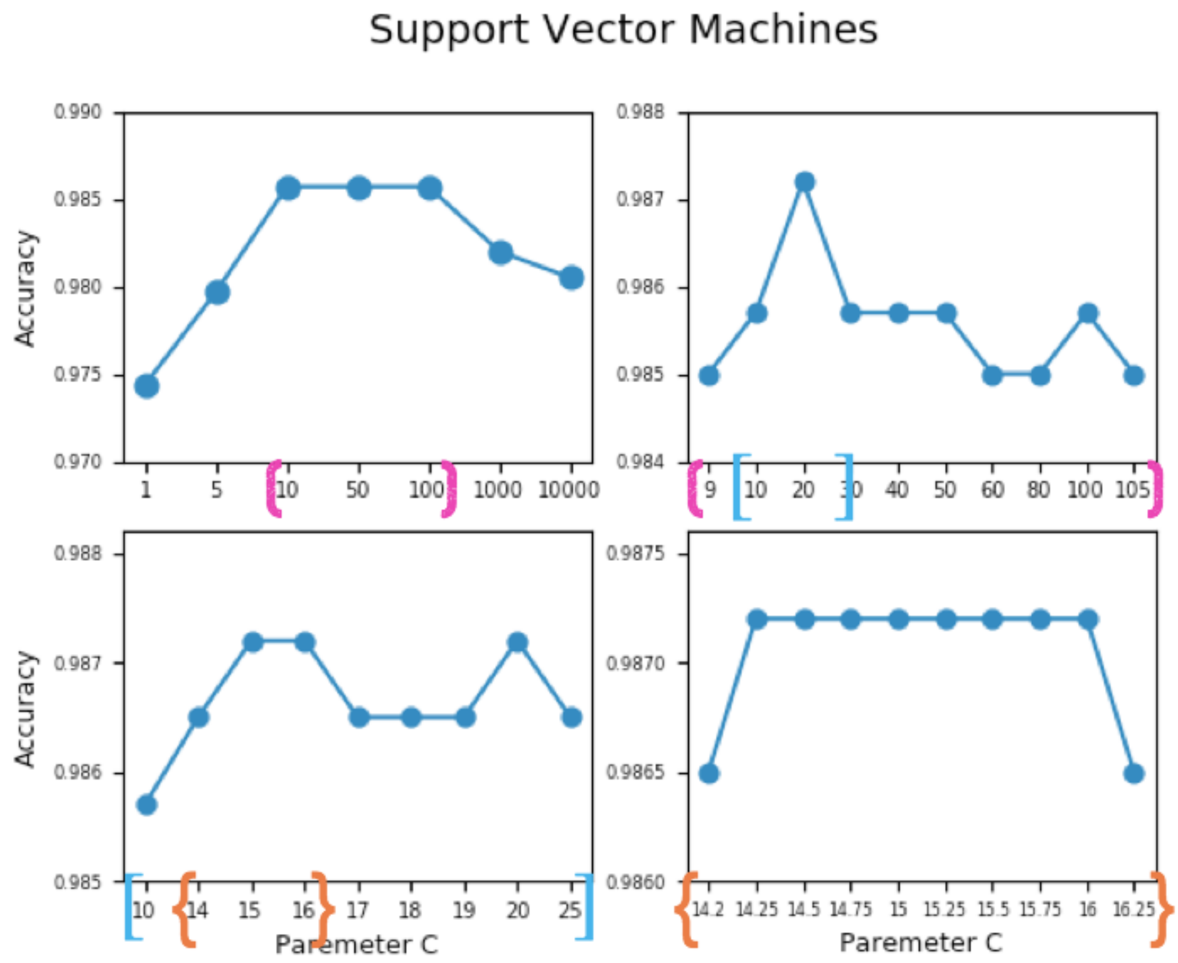


Fig. 26. Visualization of training results of SVM using different configurations.

Table 21. Classification results of the best configuration of SVM.

Category	Number of objects	Number of correctly classified objects	Number of objects classified into	Recall	Precision	F1	ROC
Beam	549	547	551	99.6%	99.3%	99.5%	99.6%
Column	286	284	286	99.3%	99.3%	99.3%	99.8%
Footing	250	250	256	100.0%	97.7%	98.8%	99.7%
Slab	53	44	46	83.0%	95.7%	88.9%	97.5%
Wall	192	186	191	96.9%	97.4%	97.1%	99.0%
Total	*1,330*	*1,300*	*1,330*	*98.6%*	*98.6%*	*98.6%*	*99.5%*

5.5.5. Random forest

Random forest uses a collection of decision trees and takes the statistical majority of the results of each tree. Search depth will determine the classification results. A low depth value may lead to underfitting, whereas a high depth value may lead to overfitting. Table 22 shows the accuracy of different configurations, a visualization of which is shown in Fig. 27, where the top one shows the trends of the accuracy, and the bottom one shows the highest value was achieved at 7 number of parents. Table 23 shows the details of the classification results of the best configuration.

Table 22. Accuracy of random forest on different number of parents.

Parents	Accuracy
1	60.08%
2	88.72%
3	94.59%
4	97.97%
5	99.17%
6	99.25%
7	**99.40%**
8	99.24%
9	99.17%
10	99.17%
11	99.17%
12	99.25%
13	99.25%
14	99.25%
15	99.25%
20	99.25%
50	99.25%
Unlimited	99.25%

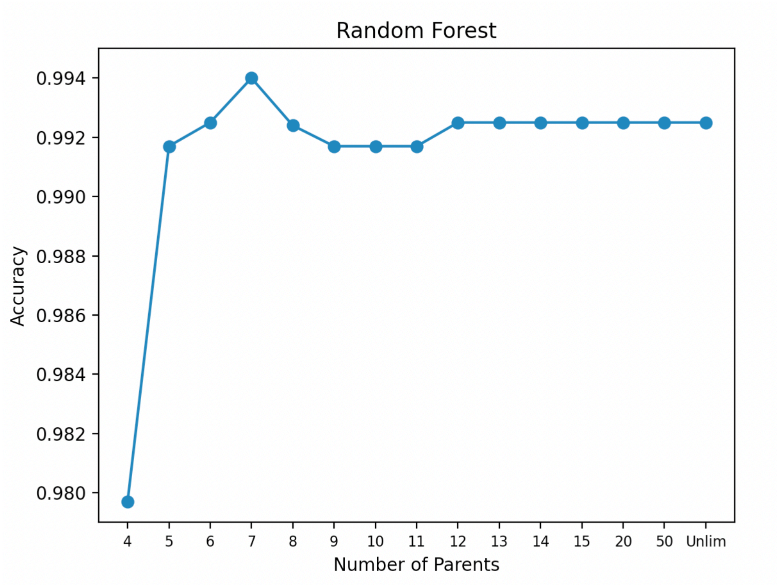


Fig. 27. Visualization of training results of random forest in different configurations.

Table 23. Classification results of the best configuration of random forest.

Category	Number of objects	Number of correctly classified objects	Number of objects classified into	Recall	Precision	F1	ROC
Beam	549	549	551	100%	99.6%	99.8 %	100%
Column	286	285	285	99.7%	100.0%	99.8%	100.0%
Footing	250	250	254	100%	98.4%	99.2%	100.0%
Slab	53	47	48	88.7%	97.9%	93.1%	99.8%
Wall	192	191	191	99.5%	99.5%	99.5%	100.0%
Total	*1,330*	*1,322*	*1,330*	*99.4%*	*99.4%*	*99.4%*	*99.9%*

5.6 Machine Learning Method Results & Analysis

5.6.1 Result of machine learning method

Random forest achieved the highest F1-measure among all the algorithms in the training phase, so the author selected random forest as the best-performing algorithm to test on the testing dataset. The overall F1-measure was 99.6% as a result. This was 0.20% higher than the training accuracy. This shows the algorithm did not overfit and perform well in the testing dataset. Table 24 shows the detailed testing results.

Table 24. Testing performance of the selected machine learning algorithm - the random forest.

Category	**Number of objects**	**Number of correctly classified objects**	**Number of objects classified into**	**Recall**	**Precision**	**F1**	**ROC**
Beam	241	241	241	100.0%	100.0%	100.0%	100.0%
Column	126	126	126	100.0%	100.0%	100.0%	100.0%
Footing	104	104	105	100.0%	99.0%	99.5%	100.0%
Slab	26	24	24	92.3%	100.0%	96.0%	99.1%
Wall	73	73	74	100.0%	98.6%	99.3%	100.0%
Total	*570*	*568*	*570*	*99.6%*	*99.6%*	*99.6%*	*99.9%*

5.6.2 Error analysis

For the best performing machine learning algorithm, the random forest, the errors were mainly due to the cutoff values of decision trees. The invariant signatures were verified to be correct, so the errors occurred because of the limitations in selected machine learning algorithm. A level of depth 7 achieved the best performance in the cross-validated data. More levels might lead to overfitting, which would reduce the training performance. The errors included 2 slabs misclassified as 1 footing and 1 wall, respectively. Fig. 28 shows the trained classifier. The tree on the left shows a branch that successfully classified all instances, which are shown in the bracket in the leaf nodes (5 columns and 15 beams were classified). The tree on the right shows a branch that only successfully classified a part of them: among the 10 objects classified as columns, 8 of them were correct and 2 of them were incorrect, while among the 18 objects classified as beams, 15 of them were correct and 3 of them were incorrect. Although each tree may make wrong predictions, e.g., in one of the branches, among 157 classified columns, 47 of them were wrong, after the voting process, the accuracy increased significantly. For example, Table 25 shows the two incorrectly classified instances. The depth of the voting tree ranged from 4 (*mRatio < 0.39; mLow < -3.901; mRatio < 0.37; Z3 >= 0.5: Footing (238/0)*) to 7 (*mLow < -7.151; ExtrudZ >= 0.5; Width >= 0.2375; X1 >= 0.9; O1 < -0.21903; Width >= 0.3178; I_R < 0.0076: Column (3/1)*).

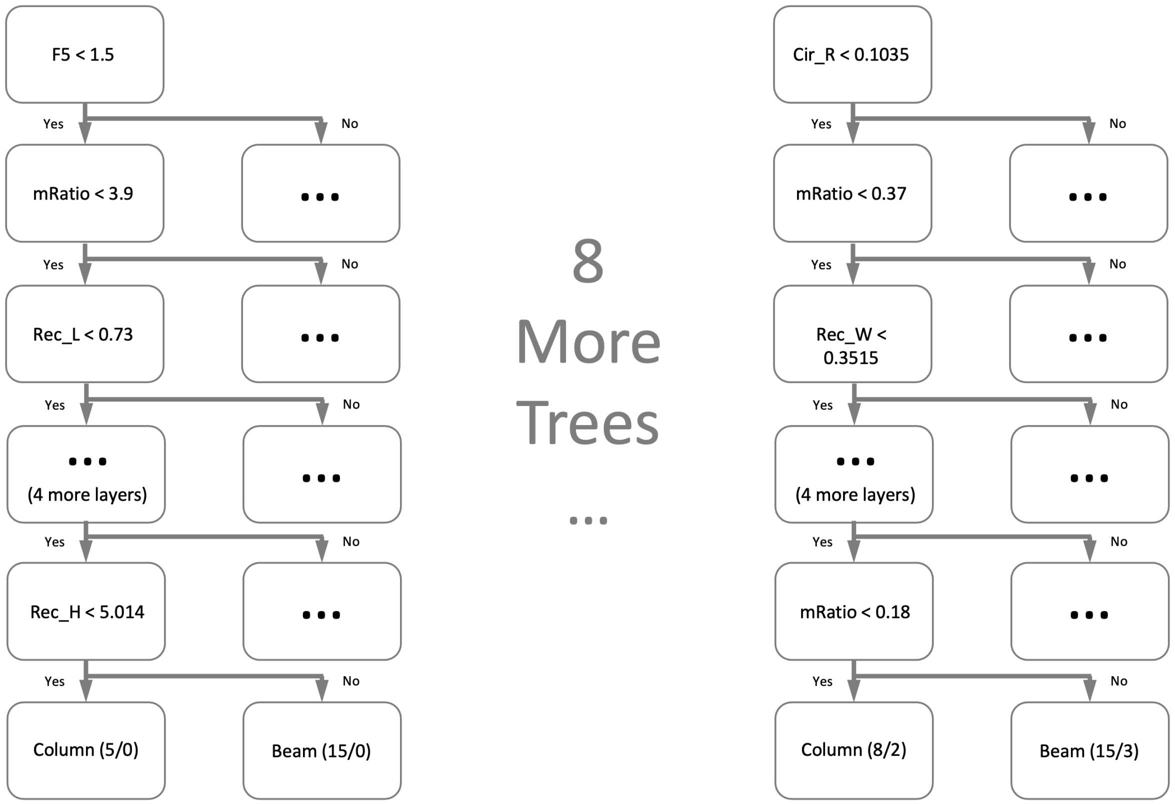


Fig. 28. Visualization of the best-performing machine learning model (random forest).

Table 25. Error analysis of the best machine learning algorithm - the random forest.

Instance	Real type	Classified type	Comment/ analysis
IfcSlab1	Slab	Footing	Voting trees: 5 footings, 3 slabs, 1 column, 1 wall
IfcSlab3	Slab	Wall	Voting trees: 5 walls, 2 footings, 2 slabs, 1 column

5.6.3 Feature set (invariant signatures) analysis

The feature set proposed in Section 45.1 led to a good performance (i.e., 99.6% F1 score in testing data) but may contain extra information that would be needed only for subcategories. After feature selection, the author proposed a 6-feature set which achieved 98.65% F1 score. The author analyzed the features as follows. The *Cir_H* feature describes the height of a cylinder shape. A zero value for this feature means a non-cylinder shape. It mainly helps distinguishing footings and columns from other object types. The *Width* feature describes the width of any shape. It is one of the general features that help distinguish many different objects. The *O3* feature describes the elevation of any object. The elevation is one of the most important pieces of information based on locational features. The *Zmin* feature describes the lowest point of faced boundary (Brep) representation, which not only tells whether the object used that representation or not, but also provides information about the size. The *ExtrudZ* feature describes the extruded direction of a swept solid representation, which not only tells whether the object used that representation or not, but also provides information about the orientation direction of the object, which is important to differentiate beams from other types.

5.6.4 Comparison with the state-of-the-art algorithm

The BIM object classification algorithm developed by Ma et al. (2018) encoded experts' insights as rules. A direct comparison would be difficult as the objects used by Ma et al. (2018) and the author were of different types. To compare with Koo et al. (2019), the author reproduced the method by Koo et al. (2019) based on the author's understanding of the feature values used by Koo et al. (2019). Because the author could not get access to the original data used by Koo et al. (2019), the author used their own dataset while extracting the features that Koo et al. (2019) proposed. After model training and parameter tuning, the author obtained 94.86% (94.12% on testing) accuracy using SVM with C=200000, which is in the same range as the experiment of Koo et al. (2019). The author also tested random forest machine learning algorithm using the same set of features and obtained 98.87% (99.12% on testing) accuracy (Table 26.).

Table 26. Performance comparison with the state of the art.

	Koo et al.'s Features	Author's Features (Invariant Signatures)
SVM	94.86% (94.12%)	98.65% (99.30%)
Random Forest	98.87% (99.12%)	99.40% (99.65%)

The random forest machine learning algorithm showed high performance in both training and testing data and is expected to be robust in similar types of BIM objects.

5.7 Rule-Based vs Machine Learning

Both rule-based method and machine learning method can be applied in solving computing problems (Chtourou and Haouari 2008) and in BIM applications (Bloch and Sacks 2018, Han et al. 2017). For this application of BIM-based AEC object classification, the comparison of the rule-based method and machine learning method is shown in Table 27.

The rule-based method achieved 100% precision and 99.1% recall, while the machine learning method achieved both 99.6% precision and 99.6% recall. As both the recall and precision were high enough, there was no need to balance the precision and recall. While both results were high, the rule-based method had a higher precision because it was less likely to make mistakes for any seen pattern. The machine learning method had a higher recall because it can automatically infer types for unseen patterns based on the features.

For efficiency, both methods were very efficient with O(1) complexity, i.e., a constant running time. As a result, both methods could generate classification results in a few seconds, which is very fast.

However, the training time for the rule-based method was significantly longer than the machine learning method, because of the human effort needed in recognizing and encoding patterns.

Table 27. Performance comparison for rule-based method and machine learning method.

	Rule-Based Method	Machine Learning Method
Precision	100%	99.6%
Recall	99.1%	99.6%
Training Time	Days	Minutes
Efficiency	O(1)	O(1)
Running Time	Seconds	Seconds

CHAPTER 6 – APPLICATIONS OF INVARIANT SIGNATURE-BASED OBJECT CLASSIFICATION

Significant portions of this chapter can be found in:

“Invariant signatures of architecture, engineering, and construction objects to support BIM interoperability between architectural design and structural analysis.” in *Journal of Construction Engineering and Management,* 147(1). DOI: 10.1061/(ASCE)CO.1943-7862.0001943.

“Constructing invariant signatures for AEC objects to support BIM-based analysis automation through object classification.” in *Journal of Computing in Civil Engineering*, submitted.

With the theoretical concept and implementation of invariant signatures, it is essential to test the robustness of the invariant signatures in solving practical BIM problems. In the previous chapter, BIM object classification was investigated. BIM object classification serves as the foundation of other BIM applications. In this chapter, based on the accurate BIM object classification, other BIM tasks, including QTO, structural analysis, and Uniformat classification are tested with the adaptation of invariant signatures and BIM object classification.

6.1 Application in QTO

Object classification is needed for a lot of common tasks in BIM. Classification accuracy is the premise for any subsequence applications. For example, QTO is an important task in BIM applications (Alshabab et al. 2017, Choi et al. 2015, Liu et al. 2016, Mandava and Zhang 2016). for QTO of wall volume, it can generate costly error if any wall object is mistakenly classified as a slab. Based on accurate QTO results, in turn, cost estimation is another important task in BIM, which therefore also relies on the accurate classification result.

Based on the automation enabled by object classification, QTO jobs can be implemented with high precision and efficiency. To demonstrate the QTO potential and verify the robustness of automated object classification based on invariant signatures and machine learning, the author tested the invariant signature-based objection classification on QTO of two types of units from a student apartment. The selected models in Fig. 29 show these two types of student apartment units

(2-bedroom unit and 4-bedroom unit) extracted from a student apartment complex (Fig. 30). As shown in Fig. 30, the units comprise of wall components and floor components. For this experiment, the author randomly selected 30 wall objects from the 4-bedroom unit and 30 wall objects from the 2-bedroom unit. The results obtained using the algorithms were compared with results of the state-of-the-art method using commercial software. The comparison of the results showed consistent values for the length, width, height, and volume of the selected objects.

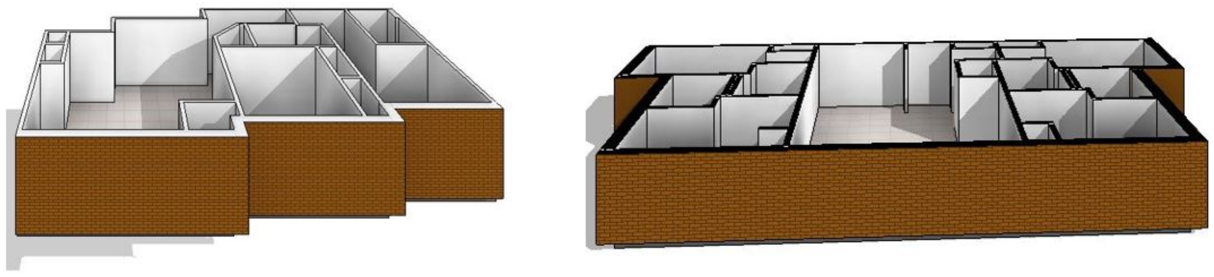


Fig. 29. Visualization of the 2-bedroom and 4-bedroom units used for quantity takeoff.

Fig. 30. Visualization of the student apartment complex used for quantity takeoff.

Table 28 shows a few sample QTO results. Table 29 shows the maximum difference, average difference, and more statistics of the QTO results.

Table 28. Example quantity takeoff results.

	Model	Volume using invariant signatures	Volume using commercial software results	Difference
Wall 1	Two-bedroom unit	149.06 ft^3	149.06 ft^3	0.0%
Wall 2	Two-bedroom unit	17.42 ft^3	14.47 ft^3	0.3%
Wall 1	Four-bedroom unit	348.03 ft^3	348.03 ft^3	0.0%
Wall 2	Four-bedroom unit	7.48 ft^3	7.48 ft^3	0.0%

Table 29. Quantity takeoff statistics.

Quantity Takeoff Statistic	Value
Largest difference	4.8%
Smallest difference	0.0%
Average of differences	0.3%
Median	0.0%
Average time of proposed method	0.079 second
Average time of commercial software	4.23 second
Time saved	98.13%
Number of objects	60

On average, the difference of the QTO results between using random forest-based object classification and using traditional approach is 0.3%, within the desirable threshold 1%. In addition, the time difference for QTO tasks was also significant. On average, each object saved 98.13% time using our proposed method. The time for QTO using the traditional approach was mainly consumed in the loading, project set-up, and element selection. This shows that the proposed invariant signature-based object classifications can produce correct QTO results that are comparable with traditional approach but with much less time, which illustrates the robustness of the invariant signature-based object classification in QTO application.

6.2 Application in Structural Analysis

The author used a data-driven approach to develop processing sub-algorithms to check and extract the geometric and material properties that include all distinguishing geometric and material information of an AEC object. Geometric properties were covered by the invariant signatures; Material properties were also used in generating material related outputs (Wu et al. 2020), but the focus of the dissertation is on the geometric parts. Table 30 shows all the needed properties related to the invariant signatures.

Table 30. Selected properties of invariant signatures.

Signature Name	Signature Type	Value Type	Feature Description
X-dim	Shape	Numerical	X dimension of bounding box
Y-dim	Shape	Numerical	Y dimension of bounding box
Z-dim (Depth)	Shape	Numerical	Extruded Depth
Extruded_direction	Shape	3D Vector	Extruded directions (Z-dim)
Origin	Locational	3D Vector	The origin of the placement
x-axis	Locational	3D Vector	Vector of x-axis
z-axis	Locational	3D Vector	Vector of z-axis

6.2.1 Iterative algorithm development overview

The author used an interative approach to develop algorithms to process an IFC model input. The workflow is shown in Fig. 31. The iterative algorithm development starts with the development of the sub-algorithms for extracting invariant signatures from all training models. Then the extracted features are verified by comparing to the original models. The extraction sub-algorithms are modified until all information are correctly extracted and verified. After verifying all the features of invariant signatures, a validating process is conducted to check if any required information is missing. In this validating process, all missing information based on geometric and material signatures are solicited until all required information are acquaired. After obtaining all the required inputs, the information mapping sub-algorithms are iteratively developed until they can successfully generate the correct results, which is based on the comparison with the predefined gold standards. In the end, all the sub-algorithms are compiled together into one processing algorithm.

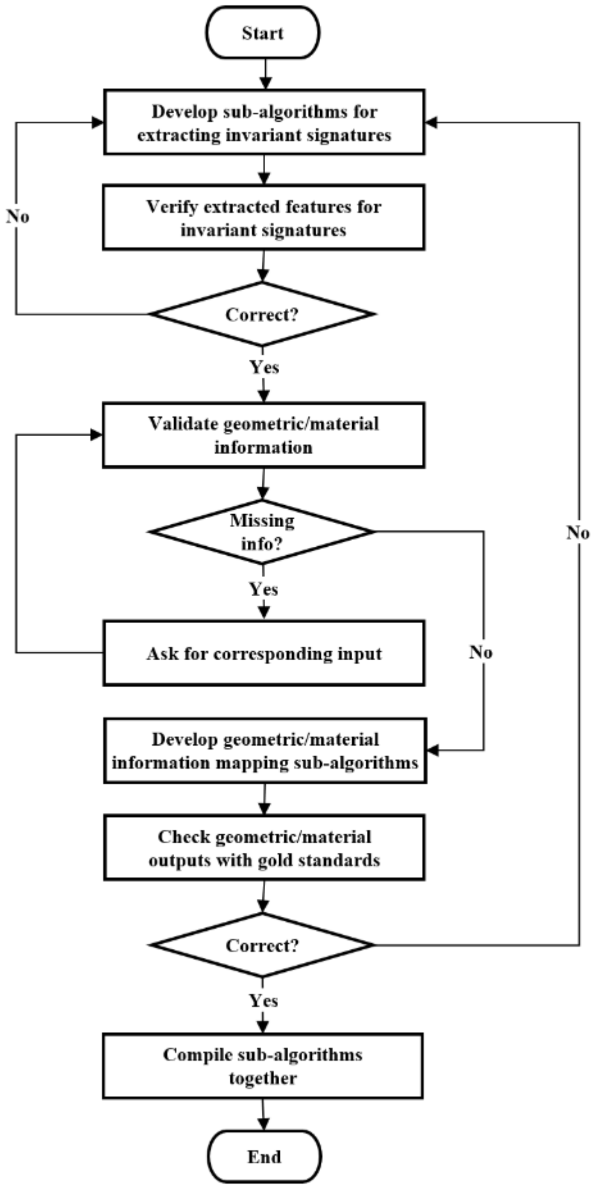

Start
Develop sub-algorithms for extracting invariant signatures
Verify extracted features for invariant signatures
Correct?
No
Yes
Validate geometric/material information
Missing info?
Yes
Ask for corresponding input
No
Develop geometric/material information mapping sub-algorithms
Check geometric/material outputs with gold standards
Correct?
No
Yes
Compile sub-algorithms together
End

Fig. 31. Iterative algorithm development.

6.2.2 Develop sub-algorithms for extracting invariant signatures & verify extracted features for invariant signatures

Started with the sub-algorithm development for extracting invariant signatures, an iterative development approach was followed until the extracted information are verified to be correct. For geometric information, both shape and locational information were extracted. All shapes in the training data were represented by either *IfcExtrudedAreaSolid* or *IfcFacetedBrep*. For

IfcExtrudedAreaSolid, objects were represented by extending a 2D plane. Regular objects can be represented using built-in 2D rectangular shapes. In this case, *X-dim* and *Y-dim* can be extracted easily because the bounding box of a rectangular cuboid is the cuboid itself. *Z-dim* (depth) and *extruded_direction* were extracted directly from the *IfcExtrudedAreaSolid.* Although all shapes in the data were represented by rectangular cuboid shape, the actual implementation of the object instances may not be regular because of the overlapping of objects, where the boundary representation (*IfcFacetedBrep*) was used. The dimensions were calculated by taking the differences between the maximum and minimum values in the *X, Y,* and *Z* dimensions. *Extruded_direction* was set vertical, i.e., using a vector of <0, 0, 1>.

For locational information, by observing the data and documentation, the author developed the sub-algorithm to calculate a resulting absolute placement from all relative placement (*IfcAxis2Placement3D*) instances associated with a given object (Fig. 32). The placement (both relative and absolute) of an object is represented by two orientation vectors (*x-vector* and *z-vector*) and a placement origin (point o). A vector or a point is represented with three values *x, y, z*, representing three dimensions. In IFC the placement of an object is usually defined in reference to the placement of another object, which forms relative placement. An absolute placement (i.e., in reference to the world coordinate) may represent the same information as multiple layers of relative placements. Using the algorithm in Fig. 32 the author was able to calculate an absolute placement *a* [(i.e., represented as new origin and axes (x-axis and z-axis)] for a series of multiple layers of relative placements by iteratively performing the merging until all placements in the list merge into one.

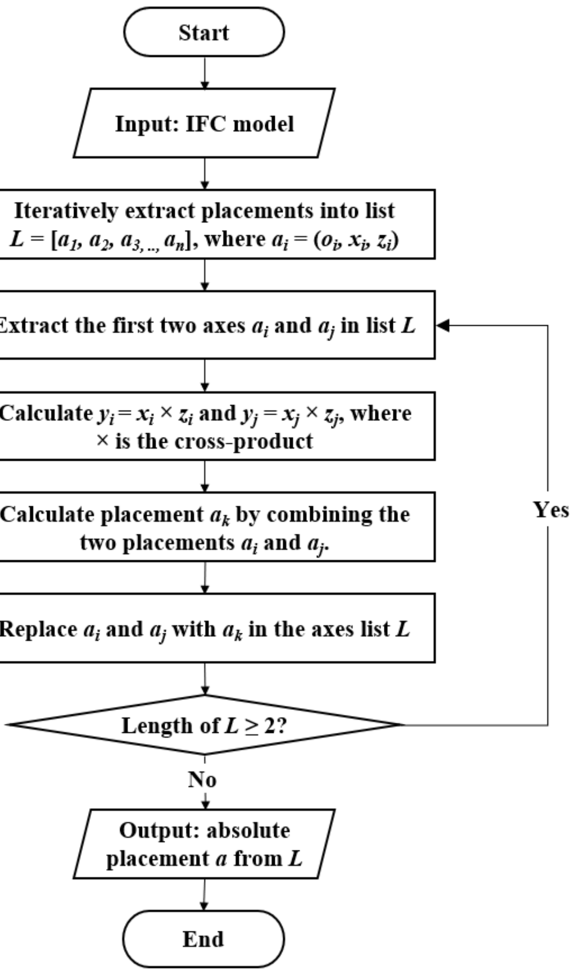

Start
Input: IFC model
Iteratively extract placements into list $L = [a_1, a_2, a_3, ..., a_n]$, where $a_i = (o_i, x_i, z_i)$
Extract the first two axes a_i and a_j in list L
Calculate $y_i = x_i \times z_i$ and $y_j = x_j \times z_j$, where $\times$ is the cross-product
Calculate placement a_k by combining the two placements a_i and a_j.
Replace a_i and a_j with a_k in the axes list L
Length of $L \geq 2$?
Yes
No
Output: absolute placement a from L
End

Fig. 32. Calculating an absolute placement that combine all relative placements.

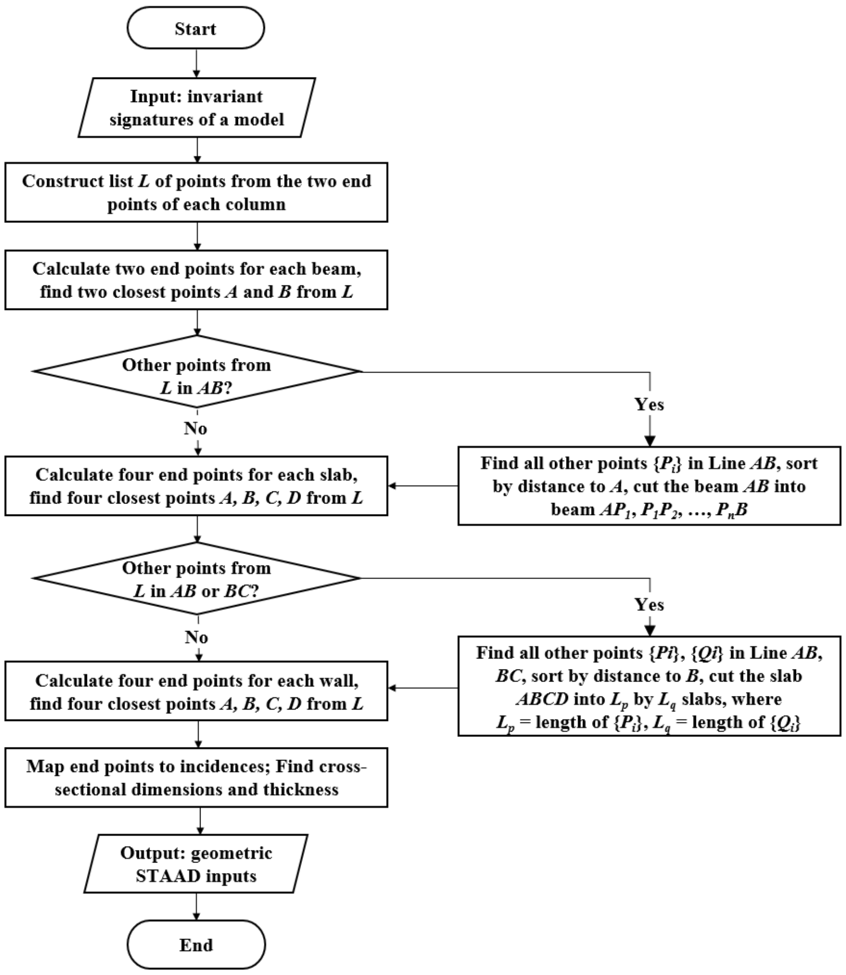
6.2.3 Validate geometric information

After feature verification, the author validated the extracted information to check if additional input were needed.

For geometric information, all needed feature values were successfully extracted. So there were no need for additional information beyond those extracted from the IFC model file.

6.2.4 Develop geometric information mapping sub-algorithms

After validating the required information in invariant signatures, the author developed information mapping sub-algorithms that convert the invariant signatures to corresponding STAAD (Bentley Company 2019) input files, i.e., a nonproprietary (textual data) structural analysis data format, for each model. Fig. 33 illustrates the developed sub-algorithm for geometric information mapping.


Start
Input: invariant signatures of a model
Construct list L of points from the two end points of each column
Calculate two end points for each beam, find two closest points A and B from L
Other points from L in AB?
Yes
No
Find all other points $\{P_i\}$ in Line AB, sort by distance to A, cut the beam AB into beam AP_1, P_1P_2, …, P_nB
Calculate four end points for each slab, find four closest points A, B, C, D from L
Other points from L in AB or BC?
Yes
No
Find all other points $\{Pi\}$, $\{Qi\}$ in Line AB, BC, sort by distance to B, cut the slab $ABCD$ into L_p by L_q slabs, where L_p = length of $\{P_i\}$, L_q = length of $\{Q_i\}$
Calculate four end points for each wall, find four closest points A, B, C, D from L
Map end points to incidences; Find cross-sectional dimensions and thickness
Output: geometric STAAD inputs
End
Fig. 33. Geometric information mapping sub-algorithm.

The generation of the STAAD inputs started from processing all the columns. Based on the required format of the STADD input file that a column object is represented by two points (Fig. 34a), which was defined as the centers of the top and bottom surfaces as the two ends of a straight-line segment. The length (YD) and width (ZD) information is generated by *X-dim* and *Y-dim* from the invariant signatures.

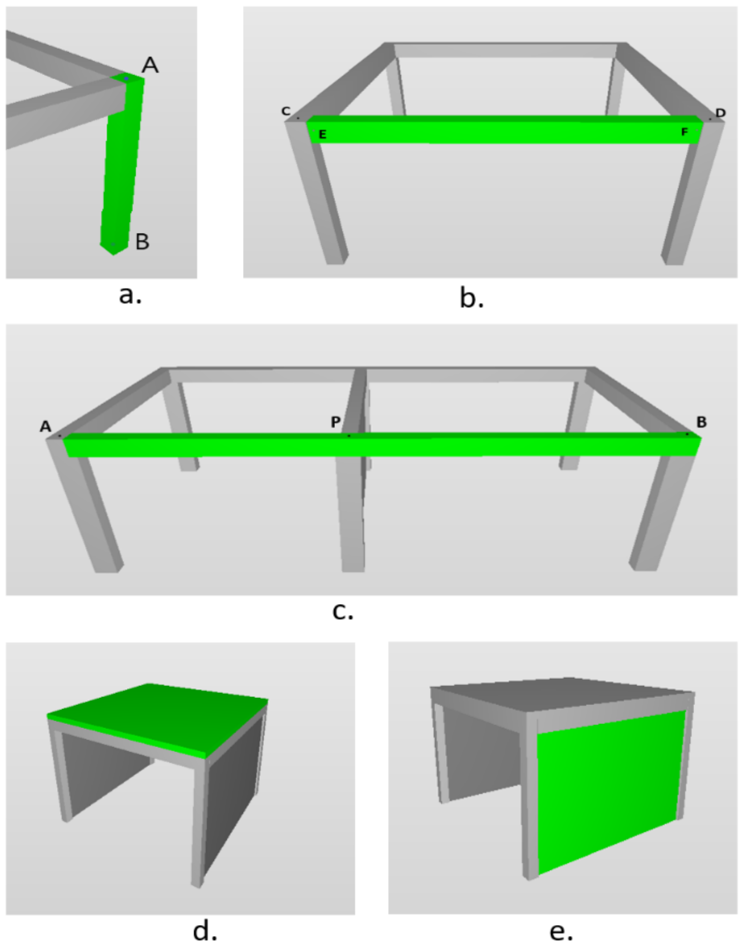


Fig. 34. Column, beam, slab, and wall examples.

To determine the coordinates of the two end points of a column (e.g., point *A* and *B* in Fig. 34a) from invariant signatures, the author observed that *B* is the origin, and *A* is calculated using Equation (1).

$$(x, y, z) = (\sigma_x, \sigma_y, \sigma_z) + (d_x, d_y, d_z) * d \qquad (1)$$

Where *(x, y, z)* is the coordinate of the end point (e.g., point *A* in Fig. 34a), *(o_x, o_y, o_z)* is the coordinate of the origin (e.g., point *B* in Figure 34a), (d_x, d_y, d_z) is the normalized extruded direction, and *d* is the extruded depth. The length and width are generated from invariant signatures to be *X-dim* and *Y-dim* or *Y-dim* and *X-dim*, based on the axis property. All end points of the columns form a list *L* that stores all the necessary points for defining all the objects.

For processing the beam instances, coordinates of the two end points were calculated from invariant signatures. Instead of directly using the coordinates of the two end points of the beam, the coordinate of the closest point of the connecting columns were used. Fig. 34b shows an example that instead of using points *E* and *F*, points *C* and *D* were used to determine the coordinates of the beam's two end points, which were the end points of the two connecting columns. The coordination of the two end points (*E* and *F*) was first calculated using Equation (1). Then the closest point of the connecting column is iteratively sought among all the column end points list *L* until the shortest Euclidian distance to point *E* or *F* is obtained. Similar to the columns dimensional definition, the length and width of a beam was generated based on *X-dim*, and *Y-dim* from invariant signatures. For the building model with multiple bays, a beam spanning two bays shall be represented by two beam members for structural analysis purpose (Fig. 34c). This scenario was captured by checking if the line of the two end points of such beam goes through another point in the *L* list. If another point *P* is in the line of *AB*, then this beam shall be split into two beams as *AP* and *PB* (Fig. 34c).

For mapping a slab information, the STAAD input file requires four end points and a numeric value for its thickness. Fig. 34d shows an example of a slab. The coordinates of the end points were calculated using Equation (2), where *origin.x*, *origin.y*, and *origin.z* represent the three coordinate values of the origin. Similar to beams, slabs may be divided into multiple slabs if the edge of a slab pass through a point in the *L* list.

$$\text{Four points} = (origin.x \pm \frac{X\_dim}{2}, origin.y \pm \frac{Y\_dim}{2}, origin.z) \qquad (2)$$

Mapping wall information was more straightforward compared to slabs, as a wall does not need to be divided into multiple walls in the multi-bay building model. The coordinates of the four end points of a wall (Fig. 34e) are calculated by Equations (3) and (4).

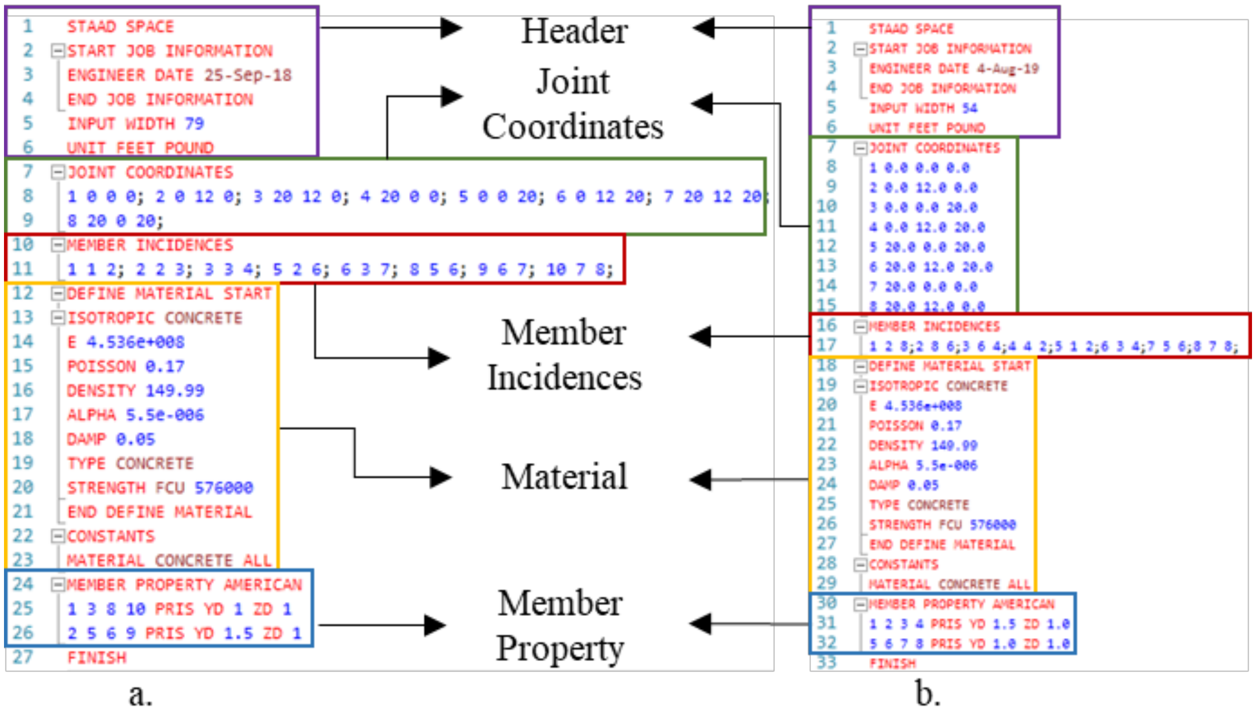
$$\text{Point 1 and 2} = (origin.x \pm max(X\_dim,\ Y\_dim), origin.y, origin.z) \ (3)$$

$$\text{Point 3 and 4} = (origin.x \pm max(X\_dim, Y\_dim)\,, origin.y, origin.z + extruded\_depth) \qquad (4)$$

Based on the invariant signatures, the author was able to identify the four points used to define a slab/wall. Then the coordinates of the columns closest to these four points were utilized to map the information of the end points of the slab/wall. The thickness was subsequently determined as the *Z-dim* for a slab object, and the minimum of *X-dim* or *Y-dim* for a wall object.

6.2.5 Check geometric outputs with gold standards

To check the processing outputs of all training models, the outputs were compared to the gold standard (i.e., the STAAD input files of the models). Sample STAAD inputs for the one bay beam-column frame developed based on invariant signatures and based on the gold standard are shown in Fig. 35a and Fig. 35b, respectively. Both input files gave exactly the same structural model in element connectivity, cross-sectional dimensions and material properties. To obtain a quantitative measure of the algorithm's performance, the author implemented metrics in describing the accuracy of each section of header, joint coordinates, member (column, beam, slab, wall) incidences, member property, material properties, and overall (Table 31). In the comparison of the joint coordinates with the gold standard, the order of the coordinates does not matter. However, they must contain the same set of coordinates. For member incidences, the comparison focused on if every object was represented with the correct ending points (incidences). Even though the data from the gold standard and from the algorithm output may appear to be different, they were considered to match if the difference was solely due to different sequences of the same incidences. In effect, the incidences define the joints connectivity based on which the length of beams, height of columns, length and width of slabs, and length and height of walls can be checked by taking the distances between the corresponding points. For beams and columns, cross-sectional dimensions were checked by YD, and ZD of MEMBER PROPERTY AMERICAN (line 24 in Fig. 35a and line 30 in Fig. 35b). For slabs and walls, the thickness was checked. The data evaluation results using the proposed metrics are summarized in Table 31, in which both the geometric and material information were generated correctly by the developed algorithm, with a 100% accuracy.



```
STAAD SPACE
START JOB INFORMATION
ENGINEER DATE 25-Sep-18
END JOB INFORMATION
INPUT WIDTH 79
UNIT FEET POUND
JOINT COORDINATES
1 0 0 0; 2 0 12 0; 3 20 12 0; 4 20 0 0; 5 0 0 20; 6 0 12 20; 7 20 12 20;
8 20 0 20;
MEMBER INCIDENCES
1 1 2; 2 2 3; 3 3 4; 5 2 6; 6 3 7; 8 5 6; 9 6 7; 10 7 8;
DEFINE MATERIAL START
ISOTROPIC CONCRETE
E 4.536e+008
POISSON 0.17
DENSITY 149.99
ALPHA 5.5e-006
DAMP 0.05
TYPE CONCRETE
STRENGTH FCU 576000
END DEFINE MATERIAL
CONSTANTS
MATERIAL CONCRETE ALL
MEMBER PROPERTY AMERICAN
1 3 8 10 PRIS YD 1 ZD 1
2 5 6 9 PRIS YD 1.5 ZD 1
FINISH
```

```
STAAD SPACE
START JOB INFORMATION
ENGINEER DATE 4-Aug-19
END JOB INFORMATION
INPUT WIDTH 54
UNIT FEET POUND
JOINT COORDINATES
1 0.0 0.0 0.0
2 0.0 12.0 0.0
3 0.0 0.0 20.0
4 0.0 12.0 20.0
5 20.0 0.0 20.0
6 20.0 12.0 20.0
7 20.0 0.0 0.0
8 20.0 12.0 0.0
MEMBER INCIDENCES
1 2 8;2 8 6;3 6 4;4 4 2;5 1 2;6 3 4;7 5 6;8 7 8;
DEFINE MATERIAL START
ISOTROPIC CONCRETE
E 4.536e+008
POISSON 0.17
DENSITY 149.99
ALPHA 5.5e-006
DAMP 0.05
TYPE CONCRETE
STRENGTH FCU 576000
END DEFINE MATERIAL
CONSTANTS
MATERIAL CONCRETE ALL
MEMBER PROPERTY AMERICAN
1 2 3 4 PRIS YD 1.5 ZD 1.0
5 6 7 8 PRIS YD 1.0 ZD 1.0
FINISH
```

Fig. 35. STAAD input: a. Gold standard. b. Results of developed algorithm.

Table 31. Algorithm evaluation results on training data (correct number of instances/total number of instances).

Section	Subsection	1	2	3	4	5
Header		6/6	6/6	6/6	6/6	6/6
Joint Coordinates		8/8	8/8	8/8	8/8	12/12
Member Incidences	Column	4/4	4/4	4/4	4/4	12/12
	Beam	4/4	4/4	4/4	4/4	12/12
	Slab	NA	1/1	1/1	1/1	2/2
	Wall	NA	2/2	2/2	2/2	1/1
Member Property	Column Dimensions	8/8	8/8	8/8	8/8	6/6
	Beam Dimensions	8/8	8/8	8/8	8/8	6/6
	Slab Thickness	NA	1/1	1/1	1/1	2/2
	Wall Thickness	NA	2/2	2/2	2/2	1/1
Material Information		8/8	8/8	8/8	8/8	8/8
Total		*46/46*	*52/52*	*52/52*	*52/52*	*78/78*

The developed algorithm was also designed to overcome the overlapping issues. The tested scenarios are: i. beams sitting on columns, ii. columns passing through beams, iii. walls cutting beams and columns, iv. slab and walls at the face of beams and columns, v. continuous slab that requires virtual cut, and vi, continuous beam that requires virtual cut. According to the results in Table 31, the developed algorithm successfully solved the overlapping issue in the training models.

6.2.6 Compile sub-algorithms together.

After checking the correctness, all the developed sub-algorithms were combined into one algorithm that can take an IFC file as an input and output a valid STAAD file to be imported by the software for structural analysis.

6.2.7 Evaluation

The algorithm developed based on the training data was then tested on the four models in the testing dataset which had different dimensions of structural configurations and members with the training models as explained previously. Similar to the training models, results of all the testing models using the developed algorithm were compared to the gold standard STAAD input files and evaluated in accuracy. According to the evaluation results (Table 32), a 100% accuracy was achieved demonstrating that invariant signature was capable of supporting the extraction of information required for conducting structural analysis including geometry, element connectivity, and cross-sectional dimensions from an IFC CV 2.0 model file created by the architectural modeling software.

Table 32. Algorithm evaluation results on testing data (correct number of instances/total number of instances).

Section	Subsection	1	2	3	4
Header		6/6	6/6	6/6	6/6
Joint Coordinates		8/8	8/8	8/8	8/8
Member Incidences	Column	4/4	4/4	6/6	8/8
	Beam	4/4	4/4	6/6	8/8
	Slab	1/1	1/1	2/2	3/3
	Wall	2/2	2/2	1/1	2/2
Member Property	Column Dimensions	8/8	8/8	12/12	16/16
	Beam Dimensions	8/8	8/8	12/12	16/16
	Slab Thickness	1/1	1/1	2/2	3/3
	Wall Thickness	2/2	2/2	1/1	2/2
Material Information		8/8	8/8	8/8	8/8
Total		*52/52*	*52/52*	*78/78*	*88/88*

In addition to the four simple testing models, the author also tested the algorithm on a building model of a real project, which is a two-story penthouse commercial building (Fig. 36 and Fig. 37). The building, for a medium size grocery store, is 31.70 meters long and 21.60 meters wide, Due to the topography limitation of the building site, the first story is a semi-basement used as storage space and the second story provides the main shopping area.

The results showed a 94.9% precision and 98.4% recall (Table 33). The author conducted an error analysis on the incorrect instances. The causes of the errors were found to include the following: a). There were 4 beams that use surface model (similar to Brep but different in the IFC representations) in the real model, where there were no such representations in the training data. So, the invariant signatures were not correctly generated. This was fixed by adding the processing of such new representations. b). The algorithm generated 5 repeated beams existing in the model, which was caused by erroneous exporting into IFC. However, this was detected and fixed by checking if the beam index already existed. c). Two slabs were not generated correctly because of their stair openings (Fig. 38), i.e., the algorithm generated the larger size of slabs because the algorithm was not trained to process openings based on training data. However, this can be addressed by finding the corresponding slab and cutting the slab with the opening to find the correct dimension. d). Column dimensions were not correct for the eight cylindrical columns. This can be fixed with the addition of radius on the dimensions. e). Four beams and five slabs were not successfully processed as they exceeded the main frame boundaries (Fig. 39). This error occurred because there were no similar situations in the training data. To fix this, the algorithms can be extended to include the addition of the objects that exceed the boundaries. For example, the algorithm can check if the ending point exceeds the boundary of the columns. If so, new end points will be added, and the same algorithm will be able to generate the desired results based on the new points. In summary, all errors were attributed to unseen patterns in the training data. As more training data is used, the algorithm can be accumulatively developed into a robust one that can successfully handle all intricate details of architectural models.

Table 33. Algorithm evaluation results on a real project (number of instances).

Section	Subsection	Generated	Correctly Generated	Total	Precision	Recall
Header		6	6	6	100%	100%
Joint Coordinates		64	64	64	100%	100%
Member Incidences	Column	44	44	44	100%	100%
	Beam	65	60	68	88.2%	92.3%
	Slab	25	23	30	83.3%	92.0%
	Wall	NA	NA	NA	NA	NA
Member Property	Column Dimensions	72	72	80	90%	100%
	Beam Dimensions	128	128	128	100%	100%
	Slab Thickness	25	25	25	100%	100%
	Wall Thickness	NA	NA	NA	NA	NA
Material Information		8	8/8	8/8	100%	100%
Total		*437*	*430*	*453*	*94.9%*	*98.4%*

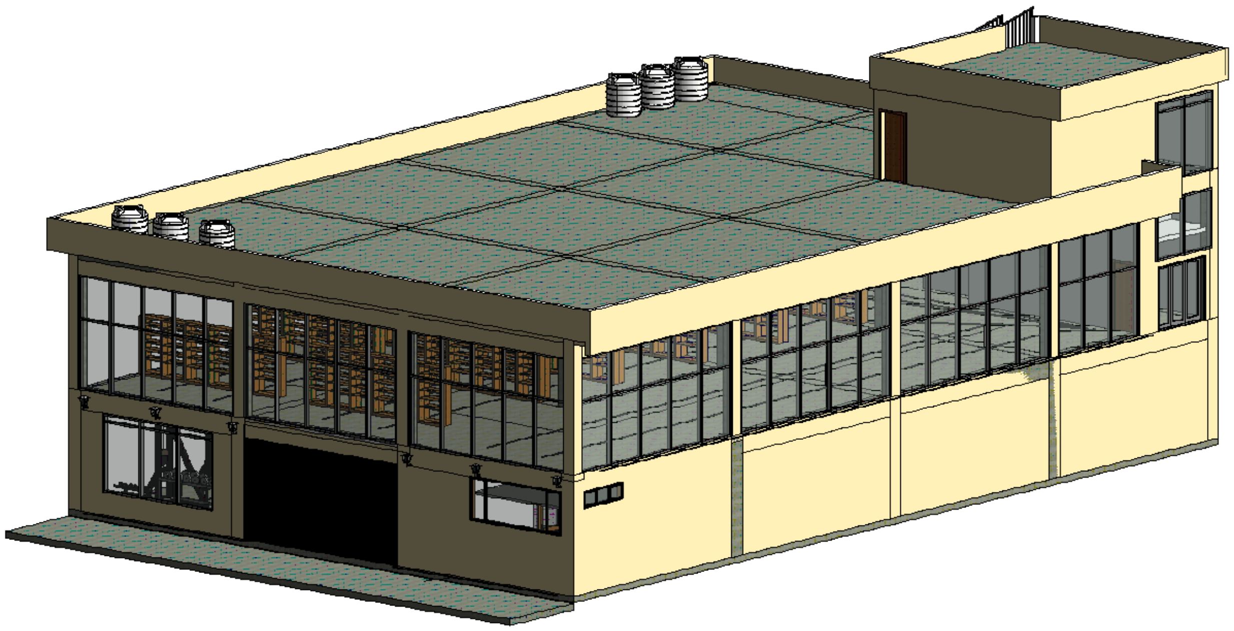


Fig. 36. Architectural view of a real project for testing.

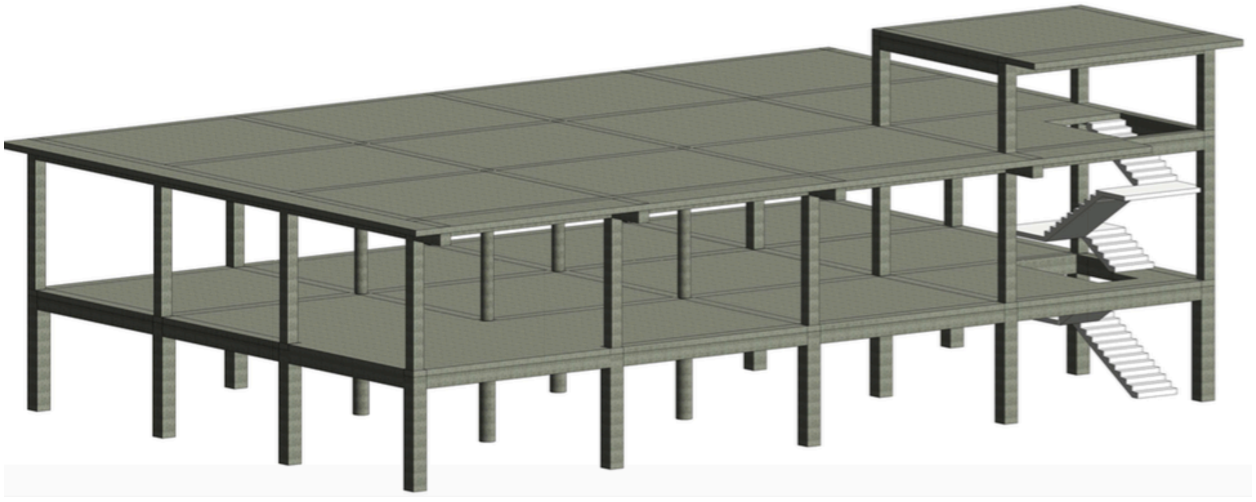


Fig. 37. Visualization of the structural view of the real project for testing.

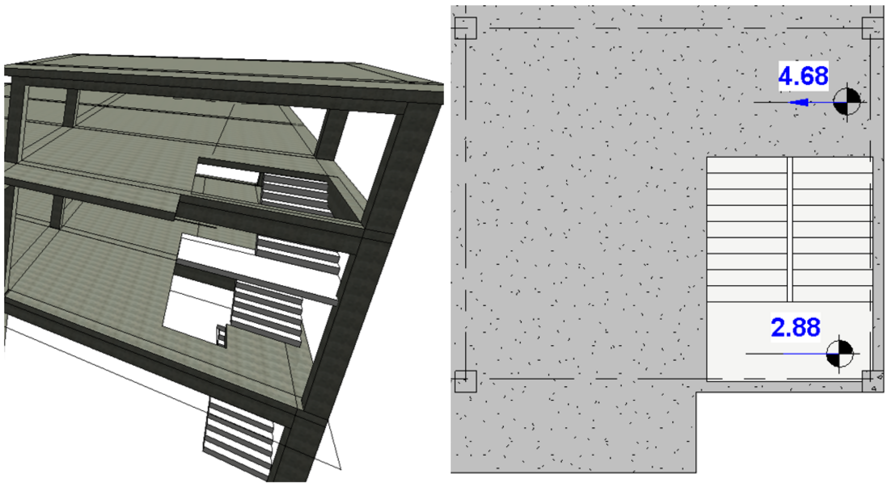


Fig. 38. The incorrectly generated slabs because of the stair openings.

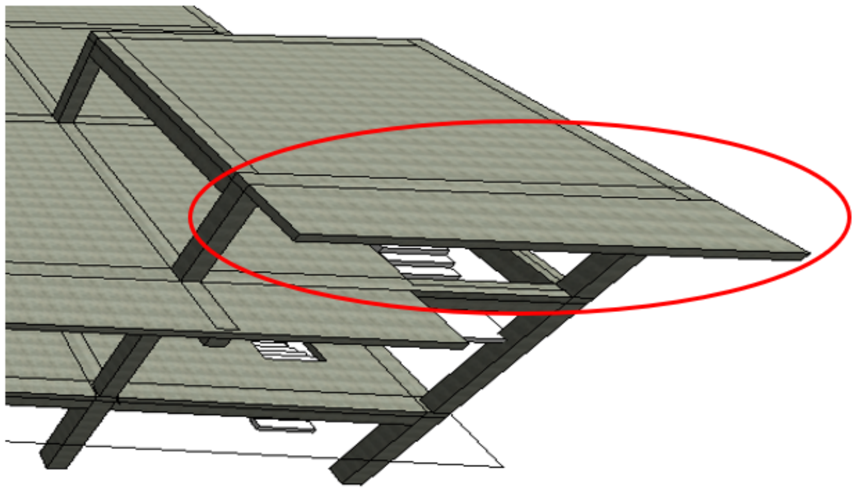


Fig. 39. An example of slabs that goes beyond the column boundary.

6.3 Application in Uniformat

Another potential use of invariant signatures for AEC objects is to support Uniformat classification. The main difference between the previous object types and the types in Uniformat is that Uniformat distinguishes exterior and interior elements, because the elements are characterized by their function with consistent economic evaluation of existing and new projects (Uniformat 2010), which is different from other classification methods. For example, Uniformat differentiates interior and exterior doors, while both of them should be represented using *IfcDoor* entities.

To illustrate the use of invariant signatures in Uniformat object classification, the author developed a proof-of-concept system to classify the AEC objects into interior and exterior doors. The algorithm was based on the egress identification algorithm developed in Chapter 7, i.e., egresses are exterior doors, and the other doors are interior doors. The results are shown in Table 34. In total, the developed algorithms were able to produce 98.7% recall and precision in recognizing interior and exterior doors.

Table 34. Performance of the developed algorithms for classifying interior and exterior doors.

Category	**Number of Objects**	**Correctly Classified**	**Classified As**	**Recall**	**Precision**
Interior Door	331	331	336	100.0%	98.5%
Exterior Door	55	50	50	90.9%	100.0%
Total	*386*	*381*	*386*	*98.7%*	*98.7%*

CHAPTER 7 – AUTOMATED BUILDING CODE COMPLIANCE CHECKING BASED ON INVARIANT SIGNATURES

A version of some parts of this chapter has been previously published as "Model validation using invariant signatures and logic-based reasoning for automated building code compliance checking." in *Journal of Computing in Civil Engineering*, submitted.

7.1 Model Validation Introduction

Fully ACC (Nguyen and Kim 2021, Yang and Xu 2004) requires accurate information extraction from both BIMs and building code chapters, and equally (if not more) importantly, a precise matching between the two. While research on information extraction has been extensively conducted for ACC, there is a lack of investigation of automated and practical information mapping between the extracted information, from BIMs to building code requirements. To address this gap, the author proposes a new method for BIMs model validation, to validate an input IFC model with regard to building code concepts. This validation method was supported by creating invariant signatures of AEC objects that capture the geometric nature of the objects. Target concepts from building codes are classified into four categories: (1) explicit concepts, (2) inferable concepts, (3) user-assisted concepts, and (4) system defaults. Identification algorithms are developed for all four categories based on the invariant signatures of AEC objects. An experiment was conducted to test the proposed method on validating five real commercial project models with selected concepts from the International Building Code 2015. Comparing to a manually developed gold standard, 99.7% precision and 99.5% recall were achieved. This demonstrates that the proposed method is promising in supporting information matching between BIMs and building code concepts for ACC purpose.

7.2 Proposed Method for ACC-Oriented BIMs Model Validation

The author proposes an iterative method to extend the information from BIMs to better support information mapping to building codes, for use in ACC systems. Different from previous approaches that directly map the building code concepts to IFC entities, which still do not provide enough information in ACC, or approaches that extend the IFC schema, which requires deep knowledge and understanding of IFC, the author proposes to use invariant signatures of AEC objects (Wu et al. 2021) as intermediate results to connect building code concepts and IFC entities, and extend the information from IFC models to infer more concepts based on the invariant signatures, to better map the information in IFC-based BIMs to building code concepts (Fig. 40).

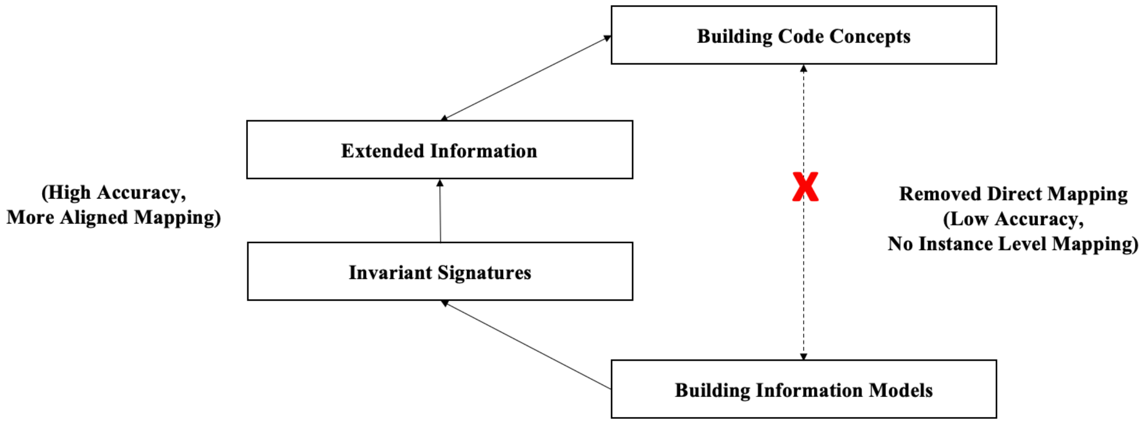

Building Code Concepts
Extended Information
(High Accuracy, More Aligned Mapping)
Invariant Signatures
X
Removed Direct Mapping (Low Accuracy, No Instance Level Mapping)
Building Information Models

Fig. 40. Idea illustration of the proposed method.

The author proposes an iterative approach to develop an intermediate information set with extended information added to BIMs, to provide information mapping between BIMs and building code concepts, which in turn better supports ACC compared to the state of the art (Fig. 41). In this proposed method, there are four main steps: (1) construct invariant signatures for AEC objects from IFC models, (2) identify and classify target concepts from logic rules that represent regulatory information from building codes, (3) develop algorithms for extending model information to match target concepts, and (4) generate logic facts to store the extended information. The algorithms are developed following a data-driven approach so that the algorithms can be continuously developed to cover more concepts and model representations. New rules and algorithms can be added under the condition that they do not interfere with previous results, to ensure compatibility and robustness. With more training data, the system will be more robust. It is not the goal in this chapter, however, to build the ultimate all-in-one set of algorithms that cover each and every possible concept in building codes. Instead, the goal in this chapter is to introduce the method and framework to enable that.

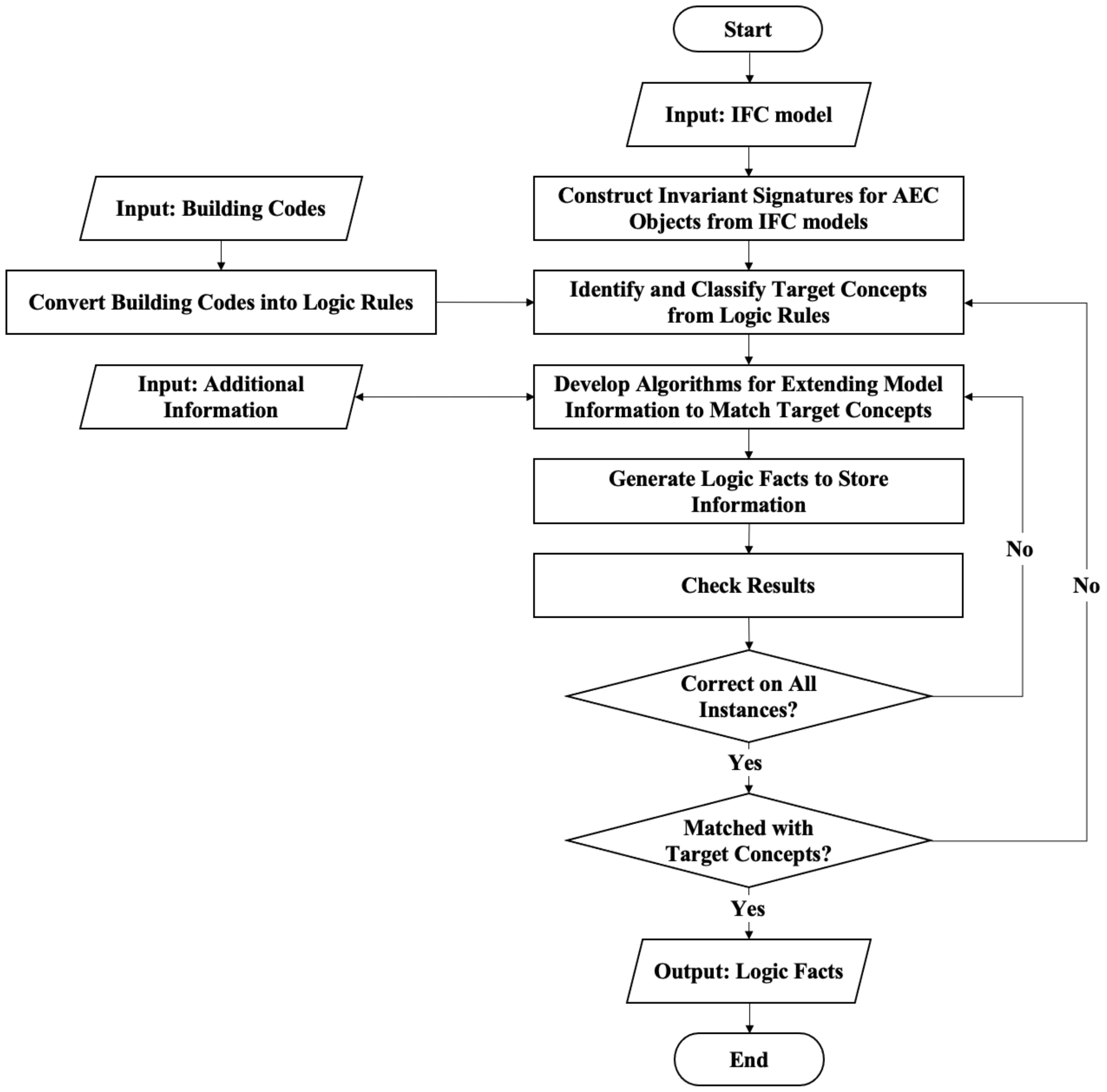

Start
Input: IFC model
Construct Invariant Signatures for AEC Objects from IFC models
Input: Building Codes
Convert Building Codes into Logic Rules
Identify and Classify Target Concepts from Logic Rules
Input: Additional Information
Develop Algorithms for Extending Model Information to Match Target Concepts
Generate Logic Facts to Store Information
Check Results
No
No
Correct on All Instances?
Yes
Matched with Target Concepts?
Yes
Output: Logic Facts
End

Fig. 41. Workflow of the proposed method.

7.2.1 Construct invariant signatures for AEC objects from IFC models

To allow the information matching in later steps, this step extracts information from the BIMs. During this processing of the BIMs data, invariant signatures (Wu et al. 2021) are used to convert the IFC models into value entries of a set of pre-defined features, a data format that is easier to process and use. Invariant signatures capture the geometric nature of the IFC model and can fully represent the AEC objects in the IFC models. In this proposed method, invariant signatures are extracted from BIMs using the IFC2x3 standard, which is commonly used in commercial projects. While invariant signatures are platform neutral and independent from specific data schema, during their use, however, they still need to be extracted from practical models that follow certain data standards. For example, consistent invariant signatures must be generated from different representations of the same shape, e.g., boundary representations (Brep) and extruded solid representations, two of the most commonly used 3D representations in IFC. In addition to these commonly used representations, a data-driven approach is used to develop algorithms for extracting information from the models that follow uncommon representations. In summary, the state-of-the-art invariant signatures extraction algorithms by Wu and Zhang (2019a, b) and Wu et al. (2021) are extended to allow full support for information extraction and information exchange for BIMs model validation.

Experimental testing requires different BIMs as training data and testing data. The performance in terms of precision and recall is reported using element-level assessment, and the results are manually verified. During the training phase, the results shall achieve 100% in information extraction from the training models, and then the trained algorithms are tested on the testing models to assess their performance.

7.2.2 Identify and classify target concepts

The target concepts are extracted from logic rules that are in turn generated from building codes (regulations) using the state-of-the-art regulatory information extraction and transformation algorithms (Zhang and El-Gohary 2015; 2016a). These algorithms can generate logic rules based on the building codes fully automatically. As the state-of-the-art algorithms are not 100% accurate yet, the generated logic rules are further manually verified and modified as necessary, to form gold standards. This is still much more efficient comparing to generating the logic rules solely manually from scratch. In the gold standard logic rules generated from the building codes, the logic clause in the form of "*entity(Entity)*" (where the predicate name has the same string with the argument but in lowercase) indicates a declaration of a concept instance. An observation of the building codes and those logic clauses showed that these predicates contain all the concepts in the corresponding sections of building codes, which are the target concepts to be mapped. On the other hand, because almost all nouns are generated in this form of instance declaration, the generated logic rules contain multiple types of building code concepts. For example, building, door opening, group h, and section 10.4 are all concepts generated from the building codes, which are of completely different types (of concepts). To allow seamless information mapping, it is critical to classify these concepts, and then develop rule-based identification algorithms based on the characteristics of each type.

In the proposed method, the author classifies the concepts into four types, which are explicit concept, inferable concept, user-assisted concept, and system default, respectively. These target concepts are classified for/by developers and end-users are not required to participate in this classification process.

Explicit concepts are the concepts that are directly generable from the BIMs, i.e., the concept has a one-to-one mapping to IFC entities and can be directly identified. For example, a wall concept has corresponding AEC objects in IFC entities: *IfcWall* and *IfcWallStandardCase*.

For non-explicit concepts, they may be identified through inference. Accordingly, there are two sub-categories, inferable and user-assisted. If a consistent inference rule can be found, then that concept is considered inferable, i.e., the inferable concepts are the concepts that can be heuristically inferred from the explicit information in the model with consistency; if not, then that concept is considered to require user judgement and therefore classified as user-assisted. For example, egress is considered an inferable concept whereas bathroom is classified as a user-assisted concept. One way to identify egress (exterior egress, all occurrence of egresses in this dissertation refers to exterior egress) concepts is by recognizing door instances at boundaries of the building that connect interior and exterior of the building. Based on the geometric and locational information, interior doors can be excluded from the egress. During this process, all the reasoning can be performed automatically without human judgment, with the developed heuristic rules. While it is not difficult for a human to judge if a door is an egress, the algorithm for identifying these requirements still needs to be developed with exact steps, to ensure the algorithm can function well to generate the expected results robustly.

In contrast, user-assisted concepts are the concepts that can be semi-inferable with additional user input and judgement. The concepts that rely solely on user input could be straightforwardly handled by taking user inputs and therefore are outside of the scope of this dissertation. While logically it is straightforward to demonstrate if a concept is inferable (i.e., by finding a way to infer it), it is close to impossible to prove if a concept is not inferable. The boundary between inferable and user-assisted concepts therefore depends on the current implementation and reasoning capability. To differentiate from inferable concepts, user-assisted concepts can be inferred to a certain extent, but not 100% inferred. For example, while it might be tempting to computationally infer bathrooms from their size (i.e., bathrooms tend to be smaller than other rooms), this cannot guarantee a consistent result, as storage rooms can also be small whereas public restaurants can have quite large bathrooms. As such, additional manual input is needed to decide if a room is a bathroom. As a result, it is hard to quantify the size of the bathrooms, and relative size does not work either. Furthermore, room functions may change during the planning phase, so additional inputs from users are required for these concepts' identification. By allowing certain extent of user judgment, the proposed method also incorporates some flexibility and error-tolerance in the building design. An important consideration is that the additional user input shall be minimal with inferred information computed to the largest extent possible, and the input should be collected preferably by multiple-choice or yes/no type of questions, with the least amount of domain knowledge and manual efforts needed.

The system defaults are the concepts that are not representing actual objects from a building, such as tables in the regulation, equations for calculations, references to other sections, or other concepts that are not directly related to BIMs, which as a result will not be the focus of this research. However, they are still needed in order for the downstream logic-based reasoning to execute successfully.

To illustrate the idea, Fig. 42 shows the relations between the four categories of target concepts based on the above-mentioned IFC models and the IBC 2015. It also reveals that direct mapping from building codes to BIMs is not feasible for all concepts, and IFC models also contain information not used in the ACC. With the extended information, such gaps in matching BIMs with building code concepts can be addressed.

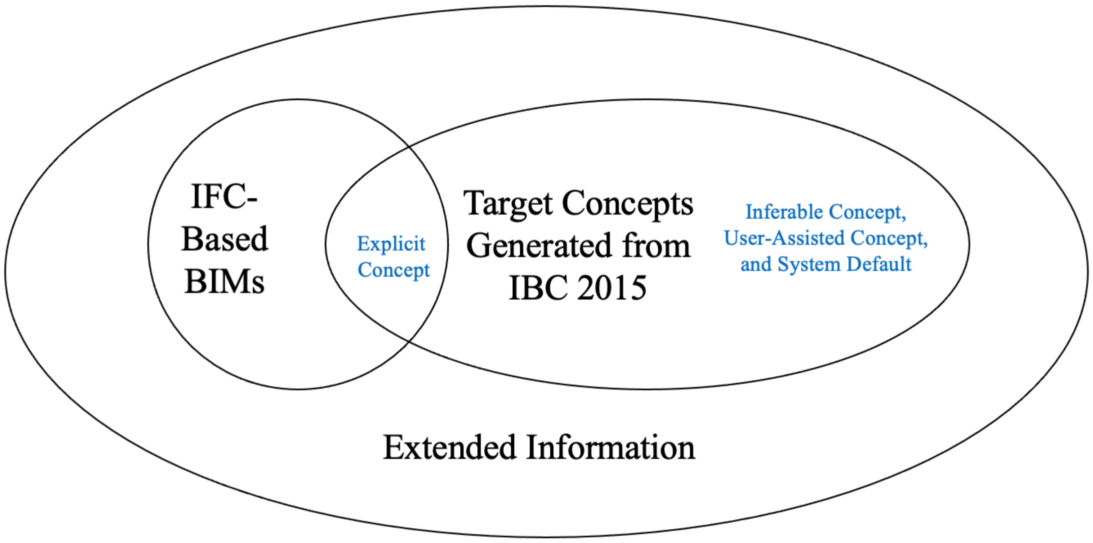

IFC-Based BIMs
Explicit Concept
Target Concepts Generated from IBC 2015
Inferable Concept, User-Assisted Concept, and System Default
Extended Information

Fig. 42. Venn diagram of concepts in building codes and IFC models.

7.2.3 Develop algorithms for extending model information to match target concepts

In this step, information extension and matching algorithms are developed to match the invariant signatures generated from Section 7.2.1 to the target concepts identified in Section 7.2.2. Heuristic rule-based algorithms are developed iteratively until consistent results are obtained in the training data, i.e., the target concepts matching need to achieve 100% precision and recall in the training data.

For explicit concepts, the algorithms are straightforward in that each object will generate one instance of the corresponding target concept. For example, a door object (IfcDoor) from the IFC model can be used to directly generate an instance of the door concept. The attributes of the instance should also be extracted, such as the length, width, etc. These attribute values can be directly obtained from the invariant signatures of the object.

For inferable concepts, heuristic rule-based algorithms are developed to extend the explicit information to the target information by inference. Each concept has its own identification sub-algorithm. For example, the egress concept (i.e., target concept) does not have a one-to-one mapping to existing IFC entities. The identification of egresses therefore needs to be conducted through inference. To develop the algorithm, heuristic rules are developed first. For example, one possible heuristic rule for egress is that doors at boundaries of the building that connect interior and exterior of the building can be recognized as egresses. An algorithm can therefore be developed to identify a door's shape, position, level, and relative locations to decide if the door can be classified as an egress, using this heuristic rule. Then all inferred results are added to the extended information set (i.e., the information originated from the BIMs), so the extended information set now contains all the inferred information in addition to the original explicit information. All information is stored for later use.

For user-assisted concepts, a light user interface (UI) is developed to allow users to input the missing information. Inference algorithms are still developed to infer intermediate results, so that the questions presented to the user only ask for minimal input, and preferably using multiple-choice questions and/or binary yes/no questions. In addition to soliciting user inputs, the algorithms also combine the user input with existing explicit and inferred information, to enrich the extended information set.

At this step, all system default concepts are identified to support further development. No further processing is needed from the model validation perspective, as the information does not directly map to the BIMs per se. For example, section_1003_3 is such a concept, which refers to Chapter 10, Section 1003.3 of a selected building code. While such concepts do not have a direct mapping in the extended information, they are important in the later automated reasoning stage of ACC.

7.2.4 Generate logic facts to store extended information set

With the extended information set produced from the previous step, logic facts are generated to store this information. Instead of higher-order logic (Gallin 2011) and defeasible logic (Naeem 2014), the logic facts store all the instances of target concepts in first-order logic (FOL) format, which is widely used and successfully tested for storing objects and relations to conduct deductive reasoning effectively and expressively (Zhang and El-Gohary 2015). It is machine-readable so that they can be directly used for logic-based ACC reasoning. The advantage of this logic fact format is that it is very easy to read, edit, store, and manage. The logic facts can be directly fed into a logic-based ACC system for automated reasoning, with no further processing needed on the model information side.

The logic facts follow a uniform representation of concepts and properties. For each instance of the target concept, a numbered instance is declared (e.g., *egress(egress32)*). Then properties of the concept are linked to the instance in the form of logic relations (e.g., *has(egress32,* *height117*) indicates the height property of egress instance number 32 is represented by height instance number 117). Different concepts may have their unique properties, e.g., the room type property applies to a room but does not apply to a window. For algorithm development of generating logic facts, each target concept uses its own designated sub-algorithm. All generated logic facts are then manually checked for evaluation.

7.3 Experiment

For experimental testing, the author collected five real commercial projects, namely, a convenience store, a warehouse, two restaurants (a fast-food restaurant and an Italian restaurant), and a hotel. All the five models were provided by our industry partner and all of them are located in Texas. All the five models, however, were designed by different architects. For each project, the author used a corresponding building model in IFC format. The author followed a 6:4 ratio for splitting the projects into training and testing models. As a result, the convenience store, warehouse, and the Italian restaurant were used as training data, whereas the fast-food restaurant and the hotel models were used as testing data. Fig. 43 shows the training projects, and Fig. 44 shows the testing projects. The hotel project contains a four-story main building and two one-story side buildings, which has the largest number of building elements, and was also largest in footage. The rest four models are all one-story buildings without side buildings. All models are well-functioned complete building models with multiple rooms, walls, doors, etc. For example, the Italian restaurant, the convenience store, the warehouse, the fast-food restaurant, and the hotel contained 14, 12, 1, 1, and 200 rooms, 34, 37, 18, 54, and 673 walls, 13, 11, 34, 11, and 317 doors, respectively. In summary, one of the testing models is similar to the three training models, which is one-story but with different layouts and different number of rooms, while the other testing model is less similar in that it contains four stories, and has significantly more rooms, which adds more complications to the validation process.

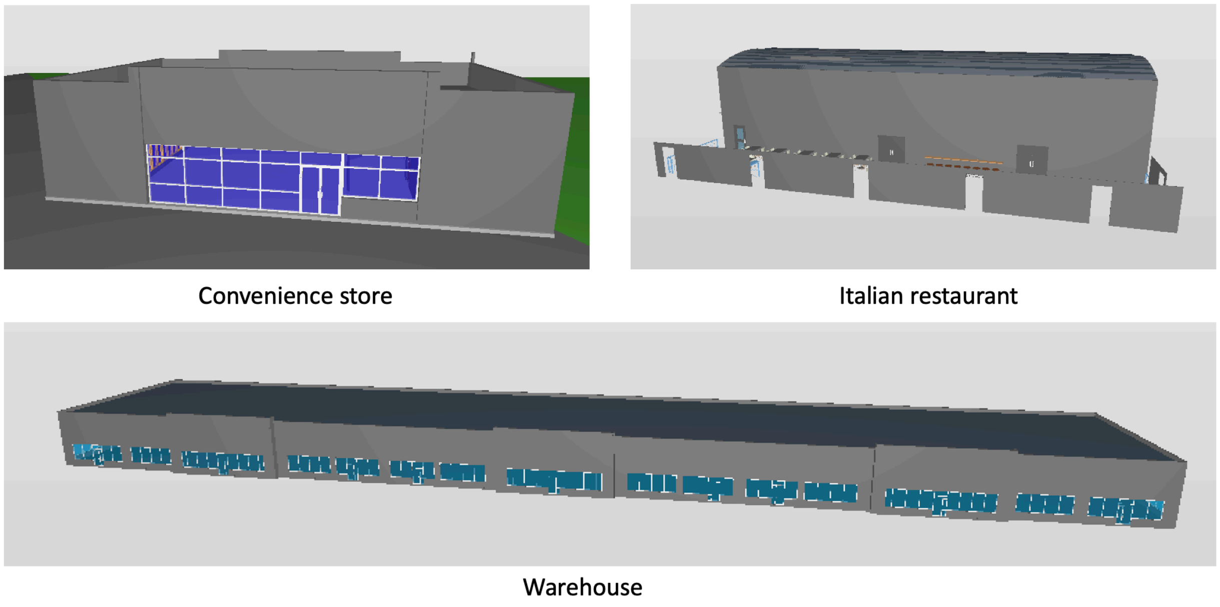


Fig. 43. Visualizations of training project models: the convenience store, the Italian restaurant, and the warehouse.

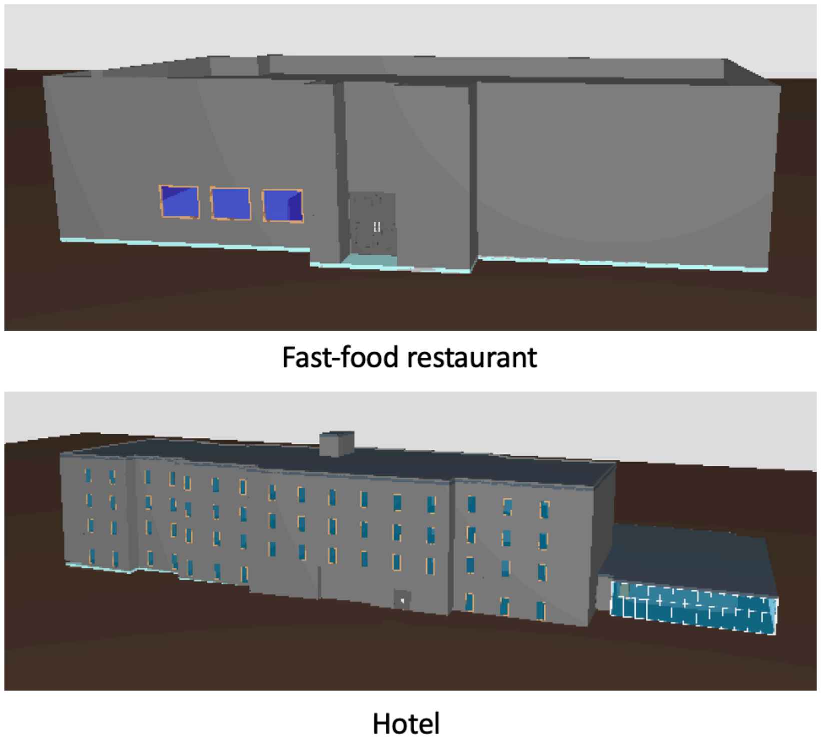


Fig. 44. Visualizations of testing project models: the fast-food restaurant and the hotel.

7.3.1 Construct invariant signatures from IFC models

For the model validation process, the first step is to preprocess the models to extract invariant signatures-based information from the IFC models.

In constructing the invariant signatures, the state-of-the-art algorithms (Wu et al. 2021) were used to extract values from the IFC models. For each AEC object, 36 features were generated as invariant signatures, including geometric, locational, and metadata signatures. The distribution of the invariant signatures-based features is shown in Table 35. The invariant signatures were able to fully represent the information about building objects of a model. This simplified the process of further conversion, as the invariant signatures were ready to be used in developing the heuristic rule-based classification algorithms. Later algorithms development follows an iterative approach, i.e., for any information found missing in the later development phase, a refinement/extension of the invariant signatures was conducted.

Table 35. Properties of invariant signatures-based features.

Signature Type	Example Feature	Feature Count
Geometric	Length, Radius	21
Locational	Position, Orientation Direction	7
Metadata	# of Faces, Average # of Vertices of Faces	8

7.3.2 Identify and classify target concepts

With the invariant signatures-based processing of IFC models completed, the building codes were then processed, using the state-of-the-art information extraction and transformation algorithm (Zhang and El-Gohary 2015; 2016b) to generate logic rules from International Building Code 2015 (IBC 2015). To illustrate the idea, the author chose Chapter 10 for demonstration. The automatically generated logic rules were then manually refined to error-freely represent the regulatory requirements in Chapter 10 of IBC 2015.

To extract the target concepts from the logic rules, the author developed a target concept identifier that can identify a target concept by recognizing the single-variable conjunct pattern of a logic clause (i.e., conjunct of the form "*entity(Entity)*"). For example, a "*door(Door)*" conjunct indicates a declaration of a door object variable. For each target concept, the logic rules always contain this type of declaration. Therefore, the target concept identification algorithm could identify all the related building code concepts from the logic rules.

With this target concept identification algorithm, the author was able to identify 1,408 target concepts. These concepts were then classified into the four types, as shown in Table 36 and Table 37. For the concept coverage of models, every model covered most of the 1408 concepts from Chapter 10 of the IBC 2015. The exceptions were fence, stair, and mezzanine. Only the Italian restaurant contained fences. Only the hotel model contained stairs. No model contained mezzanines.

Table 36. Count of concepts for each type.

Concept Type	Example Concept	Concept Count
Explicit	Wall, Ceiling, Space	8
Inferable	Egress	11
User-Assisted	Sprinkler Type	7
System Default	Equation 20, Message, Group_H	1,382

Table 37. All four types of concepts.

Explicit Concept	Inferable Concept	User-Assisted Concept	System Default
Floor	Building	Stairway_doors	Group_H
Wall	Building_height	Machinery_rooms	Means
Window	Floor_level	Storage_rooms	Barrier
Door	Floor_surface	Lobbies	Message
Room	Egress	Bathroom	1,378 more concepts
Ceiling	Assembly_areas	Exit_discharge_doors	
Fence	Means_of_egress_System	Occupied_roof	
Stair	Exit		
	Mezzanine		
	Room_elevations		
	Story		

7.3.3 Develop algorithms for extending model information to match target concepts

Explicit concepts

Explicit concepts were straightforward in their identification algorithm development, as the name suggested. Each explicit concept had a one-to-one mapping to the IFC entities, which was directly used for identifying them in the IFC models. For example, doors were identified through *IfcDoor* and rooms were identified through *IfcSpace*.

The validation of these concepts' identification was conducted by directly checking the number and properties of the corresponding IFC entities. Results showed all the needed explicit concepts were identified with 100% precision and 100% recall in the training data (Table 38).

Table 38. Explicit concepts identification result on training models.

Concept	IFC Map	No. Identified	No. Correctly Identified	Gold Standard	Precision	Recall
Floor	IfcSlab	24	24	24	100%	100%
Wall	IfcWall/IfcWall StandardCase	99	99	99	100%	100%
Window	IfcWindow	20	20	20	100%	100%
Door	IfcDoor	58	58	58	100%	100%
Room	IfcSpace	27	27	27	100%	100%
Ceiling	IfcRoof	5	5	5	100%	100%
Fence	IfcRailing	12	12	12	100%	100%
Stair	IfcStair	0	0	0	-	-
Total	-	245	245	245	100%	100%

Inferable concepts

Non-explicit concepts were more challenging in their detection comparing to the explicit concepts because there was no direct one-to-one mapping between the concepts and IFC entities. However, some non-explicit concepts could be inferred based on related information in the IFC model. The identification of these inferable concepts was therefore performed by recognizing related IFC entities and then selecting those that satisfy additional constraints based on heuristic rules. The additional constraints leveraged the geometry and relative locations of these IFC entities comparing to other IFC entities in the same model. For each inferable concept, one heuristic rule-based algorithm is developed. The algorithms are developed by analyzing the corresponding human recognition process to summarize needed heuristics, and then digitalizing and formalizing these heuristics into rules. The rules are then verified on the training data to validate the correctness of the identification algorithms during the training process.

One of the most representative inferable concepts was egress. For identifying an egress, the main heuristic was to identify the exit doors of the building. Therefore, the constraints used in the egress identification algorithm were to find doors that locate at the boundary of the building that connect interior and exterior of the building. The detailed implementation was as follows:

- Step 1: Identify the boundary of the building. The boundary is found by taking the maximum and minimum coordinates of walls and slabs at the first floor of the building. Four line segments need to be identified for each building model.
- Step 2: Select doors that are at the boundary of the building. This is achieved by checking the position of the doors. The doors selected are the ones that are close (i.e., within a predefined threshold) to the boundary of the building.
- Step 3: Verify the orientation of each door selected from Step 2 and keep the ones that connect the interior and exterior of a building. This is achieved by checking if the door's orientation is in parallel to the corresponding boundary.

The visualization of the egress identification results on the Italian restaurant model is shown in Fig. 45.

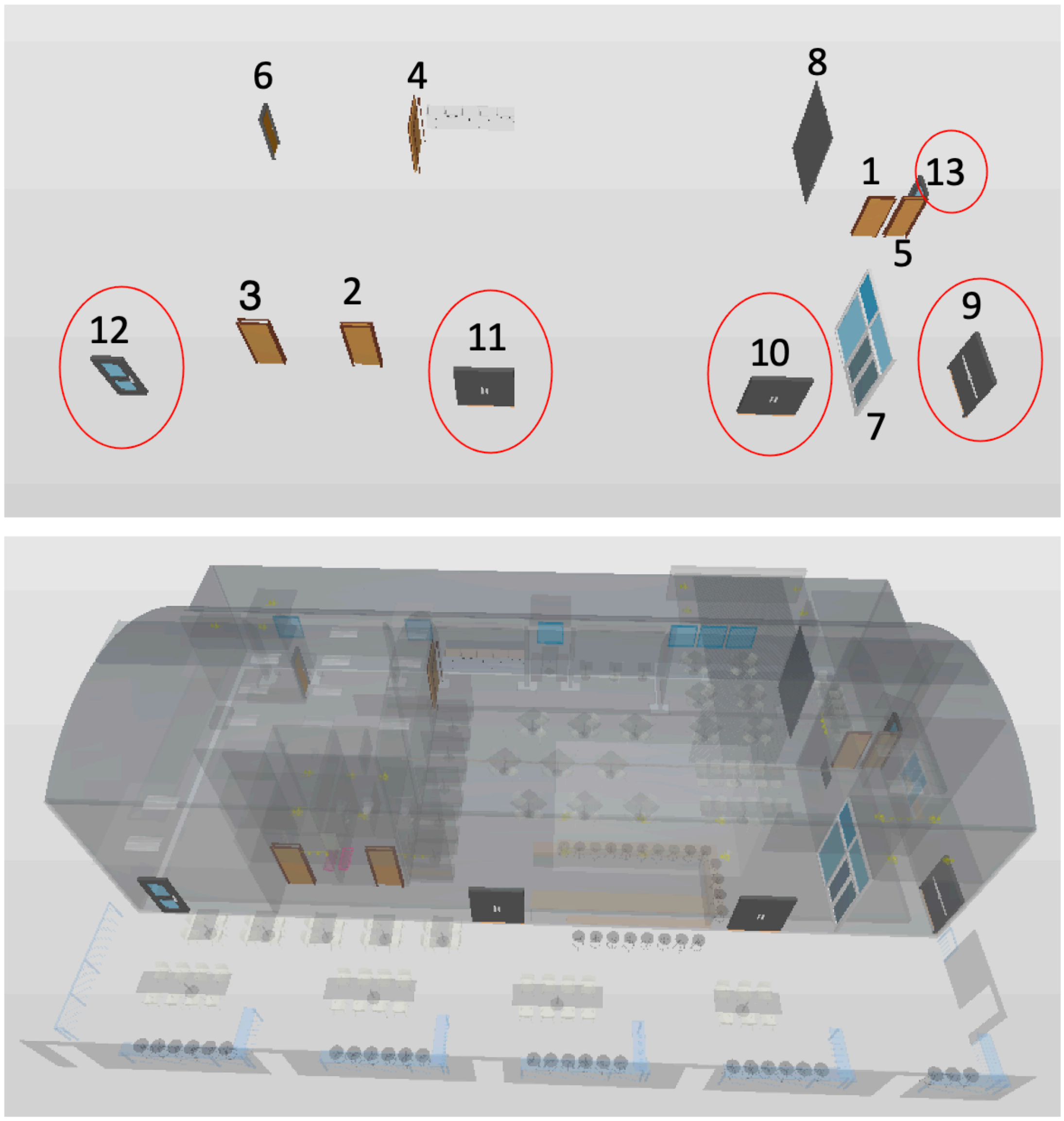


Fig. 45. Visualization of all the egresses identified for the Italian restaurant model.

As illustrated in Fig. 45, Doors 1, 2, 3, 4, 6, and 8 were not identified as egress because they were not at the boundary of the building (Step 2 criterion was not satisfied). Doors 5 and 7 were further eliminated because they failed to satisfy the orientation criterion at Step 3, i.e., the direction was not in parallel with the corresponding boundary. Finally, Doors 9, 10, 11, 12, and 13 were identified as egresses because they met all the three criteria in the algorithm.

In addition to egress, the author also developed algorithms for the rest of the inferable concepts (Table 39).

Table 39. Heuristics used in inferable concepts identification algorithms.

Concept	Heuristics Description
Building	Identify the boundary.
Building_height	One of the properties of the building concept.
Floor_level	Using clustering algorithm to identify levels.
Floor_surface	One of the properties of the floor concept.
Egress	Door entity at the boundary that connects interior and exterior.
Assembly_areas	One of the properties of the building concept.
Means_of_egress_system	Same as egress.
Exit	Same as egress.
Mezzanine	Floors objects between floor levels.
Room_elevations	One of the properties of the room concept.
Story	Sam as floor level.

The results of the algorithm development for inferable concepts are shown in Table 40.

Table 40. Inferable concepts identification results on training models.

Concept	No. Identified	No. Correctly Identified	Gold Standard	Precision	Recall
Building	3	3	3	100%	100%
Building_height	3	3	3	100%	100%
Floor_level	3	3	3	100%	100%
Floor_surface	3	3	3	100%	100%
Egress	42	42	42	100%	100%
Assembly_areas	3	3	3	100%	100%
Means_of_egress_system	42	42	42	100%	100%
Exit	42	42	42	100%	100%
Mezzine	0	0	0	-	-
Room_elevations	29	29	29	100%	100%
Story	3	3	3	100%	100%
Total	173	173	173	100%	100%

User-assisted concepts

For the other subtype of non-explicit concepts, the user-assisted concepts, the information from the IFC models was not sufficient to automatically identify those concepts even with inference. However, the inferences were able to be conducted to an extent to narrow down the range of selection. For example, based on the observation of the models, there was not enough information for differentiating bathrooms from other rooms. For the target concept of a bathroom, however, it can be inferred that it must be a room, which could help limit the range. Therefore, to identify bathrooms, the algorithm will display questions to the user to help filter through the narrowed range, e.g., "Is room2 a bathroom? Enter 1 for Yes, enter 2 for No." Similarly, algorithms for processing other user-assisted concepts were developed with interactive questions. Table 41 shows the additional information needed from users for successful identification of the user-assisted concepts. Table 42 shows the results of identifying user-assisted concepts.

Table 41. Additional information needed for user-assisted concepts.

Concept	Inferred Concept	Additional Information	Input Type
Machinery_rooms	Room	Room type.	Binary (Y/N)
Storage_rooms	Room	Room type.	Binary
Lobbies	Room	Room type.	Binary
Bathroom	Room	Room type.	Binary
Exit_discharge_doors	Egress	Egress type.	Binary
Occupied_roof	Roof	If the roof is occupied.	Binary
Stairway_doors	Door	Door types.	Binary

Table 42. User-assisted concepts identification result on training models.

Concept	No. Questions	No. Correct Questions	Gold Standard	Precision	Recall
Machinery_rooms	27	27	27	100%	100%
Storage_rooms	27	27	27	100%	100%
Lobbies	27	27	27	100%	100%
Bathroom	27	27	27	100%	100%
Exit_discharge_doors	12	12	12	100%	100%
Occupied_roof	5	5	5	100%	100%
Stairway_doors	58	58	58	100%	100%
Total	183	183	183	100%	100%

System defaults

As described above, system defaults were not representing actual objects from a building design and therefore do not have (or need to have) any mapping in the extended information set. As a result, during the model validation process, these system defaults were identified, but no further action was conducted on them. They can be used to support the later ACC stage by instantiating logic rules for automated reasoning execution. For example, an equation concept referred to an equation from the building codes. A fully ACC system would need to use the equation for calculations in the checking, the process of which should also be automated. In line with such an approach, all 1,382 system defaults concepts were identified for future use.

7.3.4 Generate logic facts to store extended information set

With all the algorithms for extending the information set, identified target concepts were stored in logic facts. The logic facts can be directly used for later logic-based ACC reasoning.

To generate logic facts, each instance of the target concept from the extended information was generated into connected B-Prolog (a Prolog system implementation with extensions for programming concurrency, constraints, and interactive graphics) clauses (Zhou 2014) including its declaration and properties, where Prolog is a FOL-based logic programming implementation. For example, the identified egress entity *1qPjXNL6r8rRF28rPA_nZx* (Door 10 in Fig. 45) was stored in "*egress(egress2). height(height32). has(egress2, height32). has_value(height32, 7.33). has_unit(height32, foot). …*" "*egress2*" was used because it was the second identified egress among all the identified egresses. The height of the egress was 7.33 feet. To represent such information, a height instance was declared as "height32", because there were already 31 height instances declared beforehand. The "*has_value(height32, 7.33)." and "has_unit(height32, foot).*" defined the value and unit of the height property, respectively. Additional information was also represented in the same format, such as its length, width, position, orientation direction, etc. Table 43 shows the results of the logic clauses generated from the extended information set.

Table 43. Number of logic clauses in the generated logic facts on training models.

Concept Type	No. Logic Clauses	No. Correct Logic Clauses	Gold Standard	Precisio n	Recall
Explicit	3,458	3,458	3,458	100%	100%
Inferable	699	699	699	100%	100%
User-assisted	80	80	80	100%	100%
System defaults	4,146	4,146	4,146	100%	100%
Total	8,333	8,333	8,333	100%	100%

7.4 Results and Analysis

To evaluate the robustness of the proposed method and developed algorithms, the author tested them on the two testing models, namely, the fast-food restaurant model and the hotel model

(Fig. 44). The results are shown in Tables 44-47. Table 44 shows the results on explicit concepts. Note that the hotel model had four roofs which were not directly observable in Fig. 44, so the author showed the roofs again in Fig. 46.

Table 44. Explicit concepts identification results on testing models.

Concept	IFC Map	No. Identified	No. Correctly Identified	Gold Standard	Precision	Recall
Floor	IfcSlab	18	18	18	100%	100%
Wall	IfcWall/IfcWall StandardCase	738	738	738	100%	100%
Window	IfcWindow	121	121	121	100%	100%
Door	IfcDoor	328	328	328	100%	100%
Room	IfcSpace	201	201	201	100%	100%
Ceiling	IfcRoof	5	5	5	100%	100%
Fence	IfcRailing	0	0	0	100%	100%
Stair	IfcStair	6	6	6	-	-
Total	-	1417	1417	1417	100%	100%

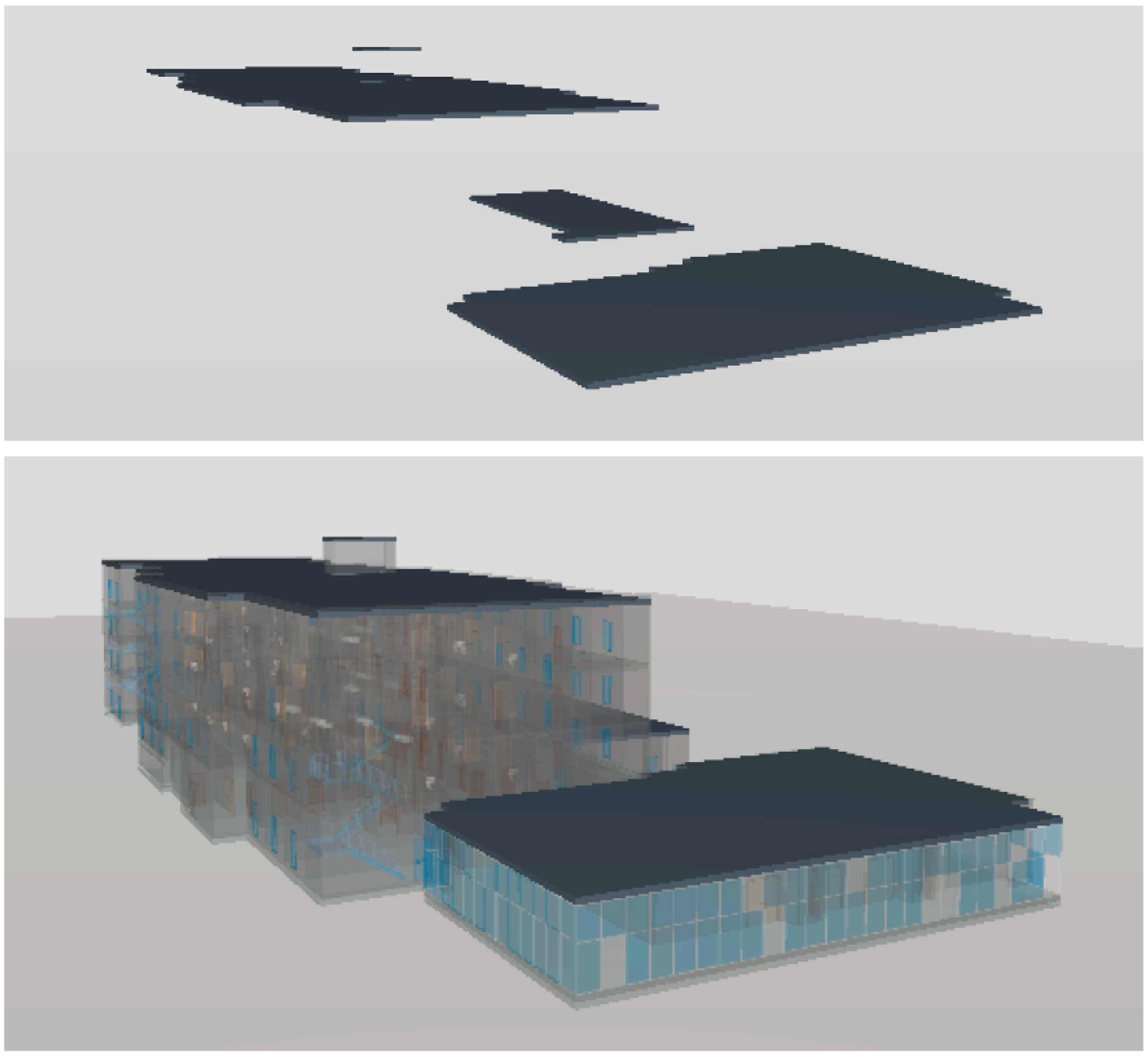


Fig. 46. Visualization of the four roofs identified for the hotel model.

Table 45 shows the results on inferable concepts. The precision was 98.1% and the recall was 94.1%. As an example illustration, the egresses identified are shown in Fig. 47 and Fig. 48, for the fast-food restaurant model and the hotel model, respectively. For fast-food restaurant model, both egresses were successfully identified. For the hotel model (Fig. 48), six egressed were successfully identified whereas five egresses were missed. This error occurred because there were three separate buildings in the hotel model, whereas the algorithm only identified one building - the main building. A further analysis showed that this was because in all three training models there was only one building for each model. Following the data-driven approach, the assumption of one building for each model was established from the training models, whereas it does not hold in the hotel testing model. As a result, the trained algorithms in our experiment failed to identify the other two buildings and the corresponding five egresses. In addition, one assembly area was also incorrectly identified because of this error.

Table 45. Inferable concepts identification result on testing models.

Concept	No. Identified	No. Correctly Identified	Gold Standard	Precision	Recall
Building	2	2	4	100%	50.0%
Building_height	2	2	4	100%	50.0%
Floor_level	5	5	5	100%	100%
Floor_surface	16	16	16	100%	100%
Egress	12	8	13	66.7%	61.5%
Assembly_areas	2	1	2	50%	50%
Means_of_egress_system	8	8	13	100%	61.5%
Exit	8	8	13	100%	61.5%
Mezzanine	0	0	0	-	-
Room_elevations	201	201	201	100%	100%
Story	5	5	5	100%	100%
Total	261	256	272	98.1%	94.1%

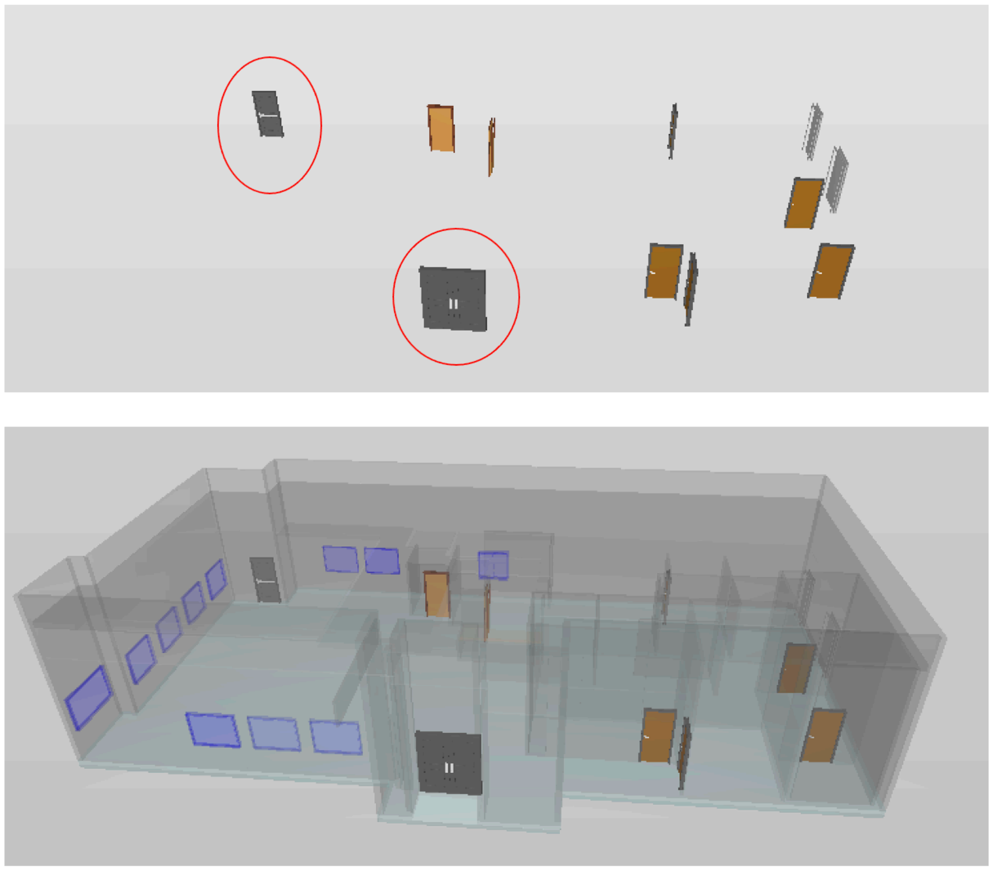


Fig. 47. Visualization of the two egresses identified for the fast-food restaurant model.

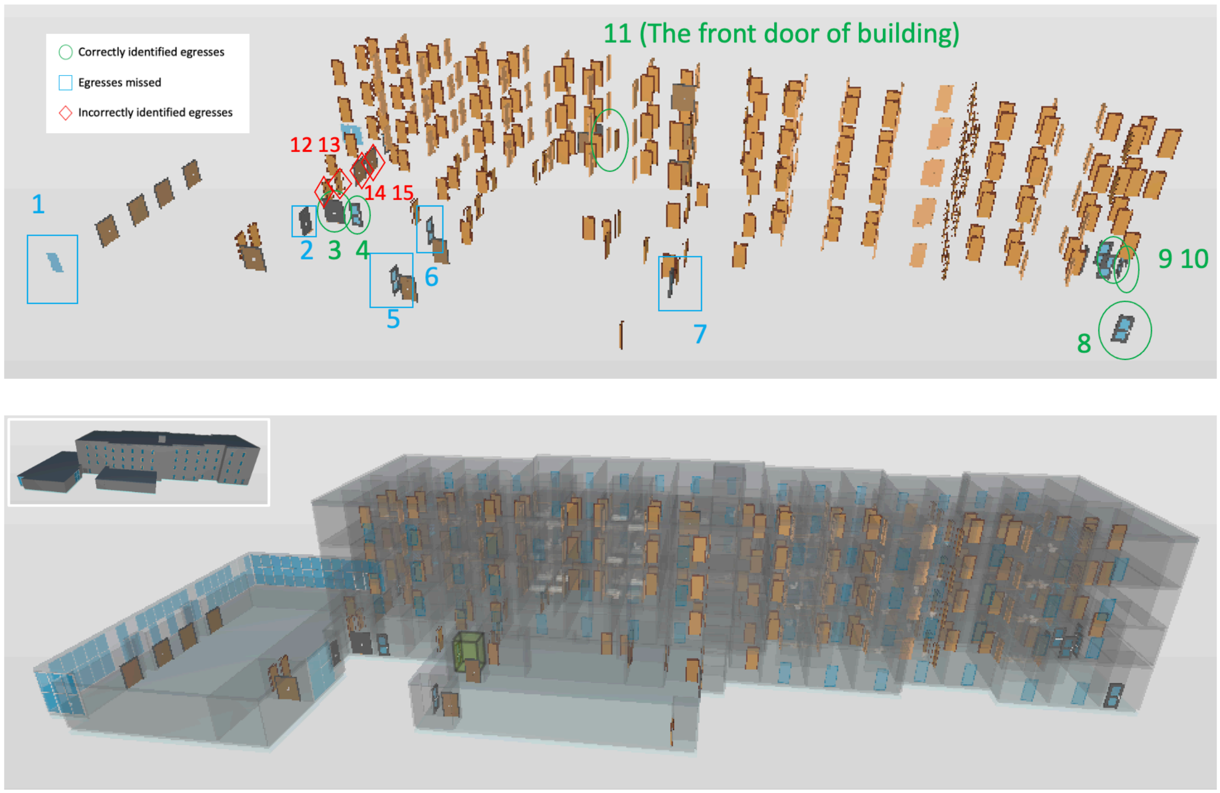

Correctly identified egresses
Egresses missed
Incorrectly identified egresses
11 (The front door of building)
12 13
14 15
1
2
3
4
5
6
7
8
9 10

Fig. 48. Visualization of all egresses identified for the hotel model (viewed from the backside of the building). Egresses 3, 4, 8, 9, 10, 11 were correctly identified, whereas egresses 1, 2, 5, 6, and 7 were missed, and egresses 12, 13, 14, 15 were incorrectly identified.

Table 46 shows the results of user-assisted concepts. The developed algorithms produced simple binary (Yes/No) questions based upon which perfect precision and recall could be achieved. Table 47 shows the generated logic facts for the whole model.

Table 46. User-assisted concepts identification results on testing models.

Concept	No. Questions	No. Correct Questions	Gold Standard	Precision	Recall
Machinery_rooms	201	201	201	100%	100%
Storage_rooms	201	201	201	100%	100%
Lobbies	201	201	201	100%	100%
Bathroom	201	201	201	100%	100%
Exit_discharge_doors	319	319	319	100%	100%
Occupied_roof	5	5	5	100%	100%
Stairway_doors	319	319	319	100%	100%
Total	1447	1447	1447	100%	100%

Table 47. Number of logic clauses in the generated logic facts on testing models.

Concept Type	No. Logic Clauses	No. Correct Logic Clauses	Gold Standard	Precision	Recall
Explicit	19,852	19,852	19,852	100%	100%
Inferable	620	555	657	88.7%	84.5%
User-assisted	70	70	70	100%	100%
System Defaults	1,382	1,382	1,382	100%	100%
Total	21,924	21,859	21,961	99.7%	99.5%

As shown in Table 47, the algorithms achieved an overall 99.7% precision and 99.5% recall which is very promising. However, for the inferable concept category, the precision was only 88.7% and recall was only 84.5% in the generated logic facts. The logic clause-level recall was lower than the concept-level recall shown in Table 45, because one egress concept needs to be represented by multiple logic clauses.

The error in the inferable concept category occurred because the developed system did not identify all three buildings in the hotel model, which can be seen in Fig. 46 and Fig. 48. A further inspection showed that the developed algorithm used a relatively strong assumption that a building is close to a rectangular shape (with some error margin) and there is only one building in the model. This assumption was valid during the training phase because all training models were single-building models with a rectangular-shaped boundary. However, this assumption was not valid on the testing hotel model because there were two side buildings attached to the main building. Therefore this hotel model did not have a one-to-one mapping between buildings and *IfcBuilding* entities (i.e., there was only one *IfcBuilding* entity but three connected buildings). This unseen multi-building model resulted in an error in the developed system and caused a 11.3% drop in precision and a 15.5% drop in recall on the logic clauses of the inferable concept category. To resolve it, the author did a follow-up experiment to modify the algorithm to allow polyline shape as building boundaries and allow multiple buildings to exist in one project model. This modification enabled the algorithm to then identify all buildings and egresses correctly.

Based on the nature of rule-based algorithms, the same level of performance may not be readily achieved on unseen patterns. However, with further development on more training data to cover more concepts and patterns for building code and building design models, respectively, the expectation is that the accumulated set of algorithms will continuously be expanded and therefore enhanced its robustness, to asymptotically approach the ultimate superset of algorithms.

In Chapter 10 of the IBC 2015, 1408 regulatory concepts can be extracted, which is sufficient to demonstrate the robustness of the method. This method is expected to generate comparable results on other chapters of the IBC 2015. For other chapters in the IBC 2015, there are many concepts that were in common with Chapter 10, such as building, wall, ceiling, etc. For new concepts, the proposed method can also be used to categorize them and develop new identification algorithms accordingly.

In summary, the proposed method achieved an overall 99.7% precision and 99.5% recall in the resulting logic clauses.

CHAPTER 8 – CONCLUSION

Significant portions of this chapter can be found in:

"Automated BIM object classification to support BIM interoperability." In *Proc., Construction Research Congress*, 706-715. DOI: 10.1061/9780784481301.070.

"New automated BIM object classification method to support BIM interoperability." in *Journal of Computing in Civil Engineering*, 33(5). DOI: 10.1061/(ASCE)CP.1943-5487.0000858.

"Invariant signatures of architecture, engineering, and construction objects to support BIM interoperability between architectural design and structural analysis." in *Journal of Construction Engineering and Management,* 147(1). DOI: 10.1061/(ASCE)CO.1943-7862.0001943.

"Constructing invariant signatures for AEC objects to support BIM-based analysis automation through object classification." in *Journal of Computing in Civil Engineering*, submitted.

"Model validation using invariant signatures and logic-based reasoning for automated building code compliance checking." in *Journal of Computing in Civil Engineering*, submitted.

"Introducing geometric signatures of architecture, engineering, and construction objects and a new BIM dataset." In *Proc., 2019 ASCE International Conference on Computing in Civil Engineering*, 264-271. DOI: 10.1061/9780784482421.034.

8.1 Conclusions

This dissertation presents a solution to solve the practical gap of lacking intuitive, uniform, and interoperable representations that can be seamlessly used for different BIM tasks. The invariant signatures, as the solution, capturing the essence of AEC objects, provide interoperable properties to represent AEC objects in BIM, which can be easily and seamlessly used in BIM tasks and applications. With this theoretical foundation, practical BIM tasks were automated seamlessly, such as AEC object classification, QTO, structural analysis, Uniformat classification, and model validation for ACC. For each BIM task, a new method was proposed to automate the process or improve the state-of-the-art performance, with the adoption of invariant signatures. As the premise of the successful experiment, a uniform and comprehensive dataset was collected.

8.1.1 Dataset

The author developed a new open BIM dataset tailored for object classification. The dataset contains five BIM models with 1,900 object instances in IFC format (and 5 additional beams as described in Section 5.2.2). The data covered the major types of AEC objects, including beams, columns, footings, slabs, and walls. The data also covered different types of representations used by the IFC entities.

The author invited independent annotators to manually label the collected data and discussed among themselves any disagreement. For objects that results could not be agreed on through discussion, the majority vote mechanism was used to decide the label to adopt. As a result, the dataset provided verified labels for the collected BIM-based AEC objects.

8.1.2 Invariant signatures

The author proposed invariant signatures of AEC objects that capture the building design information. Geometric, locational, and metadata information were included in the invariant signatures. In total, 56 invariant signatures were proposed. The invariant signatures can be extended based on needs during the development of more BIM applications.

8.1.3 BIM-based AEC object classification

This dissertation presented a data-driven, iterative method for rule-based automated classification of AEC objects in an IFC-based BIM. The method can be used to develop algorithms that read in an AEC object and automatically classify 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. An experiment was conducted on 1,905 objects 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). An algorithm was developed using the proposed method based on the training data and tested on the testing data. In common building elements categories, 99.1% recall and 100% 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 the occurrence of geometric shape representation patterns in testing data that were not included in training data. This kind 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 author's method has a constant computational complexity which is better than the linear (or higher) computational complexity of the state-of-the-art methods.

For the machine learning method, five types of machine learning algorithms were systematically investigated in AEC object classification on this dataset. The author improved the BIM object classification accuracy with the proposed invariant signatures comparing to the state of the art. In addition, the author showed that in BIM object classification, the random forest machine learning algorithm outperformed SVM and other machine learning algorithms using both the features of Koo et al. (2019) and the invariant signatures proposed by the author. The overall performance in object classification was 99.6% F1-measure. This shows that the invariant signatures and the random forest machine learning algorithm are promising in BIM object classification.

8.1.4 QTO and structural analysis

The random forest-based object classification was tested to achieve less than 1% error in QTO, compared with a gold standard developed using commercial software and a traditional manual approach, while saving significant time (98.13%) in conducting the QTO tasks.

The author used invariant signature-based object classification to propose a new data-driven method for solving practical problems in structural analysis. The method represented an effort to address the gap of the lack of foundational methods that enable a seamless BIM interoperability between architectural design and structural analysis. The developed invariant signatures, in combination with the material signatures, made the information exchange possible. An experiment was conducted to develop an information validation and mapping algorithm based on five training models, which was then tested on four testing models. Results showed that the developed algorithm successfully generated structural analysis input files for all four testing models. Comparing to an existing model conversion workflow, the proposed method achieved better accuracy. Invariant signatures were therefore demonstrated to support BIM interoperability between architectural design and structural analysis. In addition, the developed algorithm demonstrated the potential for it to be used in more complex AEC models.

The invariant signatures in combination with machine learning algorithms are expected to be applicable in a variety of other different BIM applications such as cost estimation, ACC, and building energy simulation and analysis, with high accuracy and efficiency.

8.1.5 Model validation

The author proposed an iterative method for BIMs model validation using invariant signatures and logic inference to support ACC. The proposed method can extend the information extracted from BIMs with non-explicit concepts, to support mapping from BIMs to target concepts in building codes. The proposed method uses invariant signatures of AEC objects as intermediate results to extract information from IFC entities and extended the extracted information by inferring more concepts based on heuristic rule-based algorithms and user inputs. An experiment on the International Building Code 2015 and five commercial building models showed that the proposed method achieved 99.7% precision and 99.5% recall in generating the extended information set. The proposed method was shown to function with consistent results between testing data and training data, in mapping the extended information set from BIMs to building code concepts. This helps address the existing research gap of BIMs validation to support fully ACC.

8.2 Contributions to the Body of Knowledge

8.2.1 Dataset

The author established an open dataset for BIM object classification, which can be used to reproduce the results and conduct further research. The dataset can also save time and effort of data collection and labeling for further development on the same research subject. This open and consistent data can also promote collaboration and comparison.

8.2.2 Invariant signatures

At the theoretical level, this dissertation offered an invariant signature theory with initial geometric signatures that can be used to capture the essence of AEC objects to be used in various engineering analyses and model exchange tasks. This theory goes beyond the state of the art of AEC information representation based on data schema, and enables robust AEC information representation ignorant to software implementation, modeling decisions, and/or language/cultural contexts.

In practice, this research advanced the empirical knowledge in the area of BIM interoperability by offering new data-driven methods to use invariant signatures for solving practical problems in BIM applications. This method supports the many-to-one paradigm of BIM interoperability and brings seamless and universal BIM interoperability one step closer to reality. The invariant signatures, composed of geometric, locational, and metadata signatures, were tested to be able to support AEC object classification, QTO, structural analysis, and model validation (for ACC). The invariant signatures are expected to support all other BIM activities, such as energy analysis and cost estimation.

8.2.3 BIM-based AEC object classification

The AEC object classification had both intellectual merits and practical benefits. From an intellectual perspective, this research contributes to the body of knowledge in four main ways. First, the author offered a new data-driven method for developing rule-based algorithms and a systematic machine learning method that can automatically classify IFC-based BIM objects into predefined categories. The algorithms relied on invariant signatures, i.e., the inherent geometric features, of AEC objects rather than entity or attribute names and therefore prevented classification errors caused by misuse of entities. Invariant signatures are stable and reliable properties of AEC objects (not changing with regard to software implementation, modeling decisions, an/or language/cultural contexts), and therefore object classification algorithms developed using the author's proposed methods can be more robust than those that depend on entity or attribute names.

Second, the author improved the state-of-the-art F1-measure (accuracy) for BIM-based AEC object classification from 94.9% to 99.6%, using invariant signatures as the features. The proposed method can be used by the 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.

Thirdly, a systematic approach was conducted to test and compare the rule-based algorithm and different machine learning algorithms. The best performing algorithm, random forest, was shown to outperform the rule-based algorithm and other machine learning algorithms in both the author's feature set and the state-of-the-art features.

Last but not least, the impacts of applying this work in the AEC domain could be far-reaching. The use of invariant signatures of AEC objects in classifying the objects opens a new door to BIM interoperability because such features do not change according to modelers, software providers, language, culture, or other contexts. The elimination of human-induced misuse errors in BIM enables better usability of BIM in downstream applications such as cost estimation, ACC, and structural analysis. Better usability of BIM can in turn promote the adoption of BIM in the AEC industry.

8.2.4 QTO and structural analysis

Object classification is the premise of the full automation of many BIM applications. The invariant signature-based objection classification showed that it led to comparable QTO results with the traditional approach whereas saving both time and manual effort significantly.

The research on structural analysis contributes to the body of knowledge in a unique way. The invariant signature-based structural analysis achieved better model data transfer accuracy compared to the state of the art. At the empirical level, the theoretical and practical knowledge in the area of BIM interoperability can support an improved model exchange between different AEC processes, such as between architectural design and structural analysis, between architectural design and cost analysis, and between architectural design and energy modeling and simulation.

8.2.5 Model validation

The contributions to the body of knowledge in model validation are four-fold. First, the author investigated the feasibility of automated inference of non-explicit information in BIMs, to match with building code concepts. It was found that this feasibility depends on the type of non-explicit information to be derived. For non-explicit information that is not directly inferable, the author still leveraged inference to reduce the needed manual input from users. Second, the author presented a new categorization of the concepts from building codes in four categories: (1) explicit,

(2) inferable, (3) user-assisted, and (4) system defaults, in the context of serving as target concepts of validating/matching with BIMs, to facilitate the analysis of BIMs for ACC in a divide-and-conquer manner. Third, in the identification of concepts from BIMs, the author developed invariant signatures-based algorithms for matching with each of the four categories of building code concepts, with high recall and precision. Last but not least, the proposed method was able to extend the information from BIMs and check for missing information in the context of ACC. Combined with logic rule-based algorithms, the resulting information set (i.e., the collected information) was demonstrated to be significantly extended to facilitate the matching of BIMs with building codes. In summary, the author made a solid advancement in filling the gap of information mapping from BIMs to building codes for ACC, by introducing an extended information set using invariant signatures-based inferences. This method supports model validation in ACC and therefore indirectly and positively affects interoperability of BIM.

8.3 Recommendations for Future Work.

For the BIM-based AEC object dataset, future work is recommended to expand the dataset to cover more theoretical and practical representations of AEC objects.

For AEC object classification, future work is recommended to develop the sub-algorithms with more data that belong to more categories. With a large enough dataset, the potential use of deep learning in the algorithm development phase can be investigated.

For object classification-based BIM analysis, future work is recommended to develop more robust algorithms for QTO and structural analysis with more complicated test cases and extend the application of the methods to other BIM-based tasks, such as energy analysis.

For ACC, future work is recommended to integrate the model validation method into an ACC system to perform comprehensive testing for the full ACC process.

REFERENCES

Adán, A., Quintana, B., Prieto, S.A., and Bosché, F. (2018). “Scan-to-BIM for ‘secondary’ building components.” *Advanced Engineering Informatics,* 37, 119-138.

Akanbi, T., Zhang, J., and Lee, Y.C. (2020). “Data-driven reverse engineering algorithm development (D-READ) method for developing interoperable quantity takeoff algorithms using IFC-based BIM.” *Journal of Computing in Civil Engineering*, 34(5), 04020036.

Alferes J.J. (1996). “Reasoning with Logic Programming.” Springer, Berlin Heidelberg.

Alshabab, M.S., Vysotskiy, A.E., Khali, T., and Petrochenko, M.V. (2017). “BIM-based quantity takeoff.” *Construction of Unique Buildings and Structures*, 4(55), 124-134.

Angermueller, C., Pärnamaa, T., Parts, L., and Stegle, O. (2016). “Deep learning for computational biology.” *Molecular Systems Biology*, 12(7), 878.

ANSYS, Inc (2021). “Accelerates engineering exploration, collaboration and automation.” < https://www.ansys.com> (Jul 20, 2021).

Autodesk. (2021). “Revit Sample Project Files.” < https://knowledge.autodesk.com/support/revit-products/getting-started/caas/CloudHelp/cloudhelp/2020/ENU/Revit-GetStarted/files/GUID-61EF2F22-3A1F-4317-B925-1E85F138BE88-htm.html> (Jul 20, 2021).

Azhar, S. (2011). “Building information modeling (BIM): trends, benefits, risks, and challenges for the AEC industry.” *Leadership and Management in Engineering*, 11(3), 241-252.

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 Engineering,* 3(15), 1-18.

Bethel, E.W. (2009). “Occam's razor and petascale visual data analysis.” Berkeley, CA.

Beleites, C., Neugebauer, U., Bocklitz, T., Krafft, C., and Popp, J. (2013). “Sample size planning for classification models” *Analytica Chimica Acta*, 760, 25-33.

Bloch, T., and Sacks, R. (2018). “Comparing machine learning and rule-based inferencing for semantic enrichment of BIM models.” *Automation in Construction*, 91(2018), 256-272.

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.

Brandman, T. and Peelen, M. (2017). “Interaction between scene and object processing revealed by human fMRI and MEG decoding.” *The Journal of Neuroscience*, 37(32), 7700-7710.

Coolidge, J.L. (1902) “Euclid's elements.” *Bulletin of the American Mathematical Society*, 8(5), 216-220.

STAAD (2019). Bentley Company. < https://www.bentley.com/en/products/brands/STAAD> (Nov. 2019).

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).

buildingSMART. (2021). “IFC specifications database.” <https://technical.buildingsmart.org/standards/ifc/ifc-schema-specifications/> (Apr 22, 2021).

Cartwright, H. (2015). “Artificial neural networks.” Springer*,* New York.

Cartwright, H. (2008). “Using artificial intelligence in chemistry and biology: a practical guide.” CRC Press, Boca Raton, FL.

Chen, J., Fang, Y., and Cho, Y. (2016). “Automated equipment recognition and classification from scattered point clouds for construction management applications.” *Proc. International Symposium on Automation and Robotics in Construction (ISARC)*, Waterloo, Auburn, AL, 33(1).

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

Choi, J., Kim, H., and Kim, I. (2015). “Open BIM-based quantity takeoff system for schematic estimation of building frame in early design stage.” *Journal of Computational Design and Engineering,* 2(2015), 16-25.

Chtourou, H. and Haouari, M. (2008). “A two-stage-priority-rule-based algorithm for robust resource-constrained project scheduling.” *Computers and Industrial Eng*ineering, 55(1), 183-194.

Cohen, H, and Lefebvre, C. (2017). “Handbook of categorization in cognitive science.” Elsevier Ltd., 2nd edition, Netherlands.

Daniyarova, E., Myasnikov, A., and Remeslennikov, V. (2012). “Algebraic geometry over algebraic structures. V. The case of arbitrary signature.” *Algebra and Logic*, 51(1), 28-40.

Dimyadi, J., and Henderson, S. (2012). “Open IFC Model Repository.” < http://openifcmodel.cs.auckland.ac.nz/> (Feb. 13, 2019).

Ding, L. Drogemuller, R., Rosenman, M., Marchant, D., and Gero. J. (2006). “Automating code checking for building designs: DesignCheck.” *Clients Driving Innovation: Moving Ideas into Practice*, 1-16.

Ding, L., Li, K., Zhou, Y., and Love, P.E.D. (2017). “An IFC-inspection process model for infrastructure projects: Enabling real-time quality monitoring and control”. *Automation in Construction*, 84, 96-110.

Eastman, C., Lee, J., Jeong, Y., and Lee, J. (2009). “Automatic rule-based checking of building designs.” *Automation in Construction*, 18(8), 1011-1033.

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.

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

ETakeoff, LLC (2021). “The next evolution in electric takeoff” <https://etakeoff.com/> (Jul 20, 2021).

Fang, Y., Chen, J., Cho, Y., and Zhang, P. (2016). “A point cloud-vision hybrid approach for 3D location tracking of mobile construction assets.” *Proc.*, *International Symposium on Automation and Robotics in Construction (ISARC)*, Auburn, AL.

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

Feng, C., Liu, M., Kao, C., and Lee, T. (2017). "Deep active learning for civil infrastructure defect detection and classification." *Proc. Computing in Civil Engineering,* ASCE, Reston, VA, 298-306.

Fernández, M., Cantador, I., López, V., Vallet, D., Castells, P., and Motta, E. (2011) "Semantically enhanced information retrieval: an ontology-based approach." *Web Semantics*, 9(4), 434-452.

Fenves, S. J. (1966). "Tabular decision logic for structural design." *Journal of the Structural Division*, 6(92), 473-490.

Fieldwire. (2021). "BIM viewer." < https://www.fieldwire.com/bim/> (Jul 20, 2021).

Gallaher, M.P., O'Connor, A.C., Dettbarn, J.L., Jr., and Gilday, L.T. (2004). "Cost analysis of inadequate interoperability in the U.S. capital facilities industry." NIST GCR 04-867, National Institute of Standards and Technology, Gaithersburg, MD.

Garrett, J. H., Jr. and Fenves, S. J. (1987). "A knowledge-based standard processor for structural component design." *Engineering with Computers*, 2(4), 219-238.

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

General Service Administration (GSA). (2007). "BIM guide overview." *GSA BIM Guide Series 01.*, U.S. General Services Administration, Washington, DC. <http://www.gsa.gov/portal/getMediaData?mediaId=226771> (Feb. 13, 2019)

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." *Journal of Computing in Civil Engineering,* 29(1).

Gopalakrishnan, 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." *International Journal for Traffic and Transport Engineering,* 8(1), 1-14.

Grégoire, G. (2014). "Logistic regression." *EAS Publication Series*, 66, 89-120.

Guillaumin, M., Küttel, D., Ferrari, V. (2014). "ImageNet auto-annotation with segmentation propagation." *International Journal of Computer Vision*, 110(3), pp.328-348.

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).

Hamledari, H.,McCabe, B., and Davari. S. (2017). “Automated computer vision-based detection of components of under-construction indoor partitions.” *Automation in Construction,* 74, 78-94.

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

Han, P., Siu, M., AbouRizk, S., Hu, D., and Hermann, U. (2017). “3D model-based quantity takeoff for construction estimates.” *Proc., ASCE International Conference on Computing in Civil Engineering,* ASCE, Reston, VA, 118-124.

Hastie, T., Tibshirani, R., and Friedman, J. (2009). “The elements of statistical learning: data mining, inference, and prediction.” Springer*,* New York.

Haubler, M., Esser, S., and Borrmann, A (2021). “Code compliance checking of railway designs by integrating BIM, BPMN and DMN.” *Automation in Construction,* 1, 121.

Hébert, A.(2016). “DRAGON5 and DONJON5, the contribution of École Polytechnique de Montréal to the SALOME platform.” *Annals of Nuclear Energy*, 87(1), 12-20.

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.

Henn, A., C. Römer, G. Gröger, and L. Plümer. 2012. “Automatic classification of building types in 3D city models.” *GeoInformatica,* 16 (2): 281–306.

Hjelseth, E. and N. Nisbet (2011). “Capturing normative constraints by use of the semantic mark-up (RASE) methodology.” *Proc., CIB W78 2011 - 28th International Conference-Applications of IT in the AEC Industry*, Conseil International du Bâtiment (CIB), Rotterdam, Netherlands.

Holzer, D. (2015). “BIM manager's handbook. Best practice BIM. EPart 1.” Wiley, West Sussex, England.

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 Engineering Research and Innovation Conference*, Atlanta, GA.

Hussain, K., and Choudhry, R.M. (2013). “Building information modeling (BIM) uses and applications in Pakistan construction industry.” *Proc. 13th International Conference on Construction Applications of Virtual Reality*, 30-31 London, UK.

International Organization for Standardization (ISO). (2013). “ISO 16739:2013 Industry Foundation Classes (IFC) for data sharing in the construction and facility management industries.” International Organization for Standardization, Geneva, Switzerland. <https://www.iso.org/standard/51622.html.> (Feb. 13, 2019)

Jensen, F. (1996). “An Introduction to Bayesian Networks”. Springer. Germany.

Jeong, J., and Lee, G. (2009). “Requirements for automated code checking for fire resistance and egress rule using BIM.” Proc., *International Conference on Construction Engineering and Project Management*, 316-322.

Johnson, J.S., and Olshausen, B.A. (2003). “Timecourse of neural signatures of object recognition.” *Journal of Vision*, 3(7), 499-512.

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 Engineering,* 118, 1060-1067.

Kasim, T., Li. H., Rezgui, Y., and Beach, T. (2013). “Automated sustainability compliance checking process: proof of concept.” *CONVR 2013*, <http://itc.scix.net/paper/convr-2013-1> (Feb. 1 2021).

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.

Khemlani, L. (2011). “CORENET e-PlanCheck: Singapore’s automated code checking system. AECbytes.” <http://www.aecbytes.com/feature/2005/CORENETePlanCheck.html> (accessed 2/1/2021).

Kim, H., Shen, Z., Kim, I., Kim, K., Stumpf, A., and Yu, J. (2016). “BIM IFC information mapping to building energy analysis (BEA) model with manually extended material information.” *Automation in Construction*, 68, 183-193.

Koo, B., and Shin, B. (2018) “Applying novelty detection to identify model element to IFC class misclassifications on architectural and infrastructure building information models.” *Journal of Computational Design and Engineering*, 5(4), 391-400.

Koo, B., La, S., Cho, N.W., and Yu, Y. (2019). "Using support vector machines to classify building elements for checking the semantic integrity of building information models." *Automation in Construction*, 98, 183-194.

Kotsiantis, S. B. (2007). "Supervised machine learning: a review of classification techniques." *Informatica*, 31(3), 249(20).

Kuang, D., and Xu, B. (2018) "Predicting kinetic triplets using a 1d convolutional neural network." *Thermochimica Acta*, 669, 8-15.

Langley, P. (1994). "Selection of relevant features in machine learning." Institute for the study of learning and expertise Palo Alto, California. <https://apps.dtic.mil/dtic/tr/fulltext/u2/a292575.pdf > (Feb. 13, 2019)

Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C.L. (2014). "Microsoft COCO: common objects in context." *Lecture Notes in Computer Science*, 8693(5), 740-755.

Liu, H., Lu, M., and Al-Hussein, M. (2016). "Ontology-based semantic approach for construction-oriented quantity take-off from BIM models in the light-frame building industry." *Advanced Engineering Informatics*, 30 (2016), pp 190-207.

Liu, K., and Golparvar-Fard, M. (2015). "Crowdsourcing construction activity analysis from jobsite video streams." *Journal of Construction Engineering and Management,* 141(11), 4015035.

Lopez, L. A. and Wright, R. N. (1985). "Mapping principles for the standards interface for computer aided design." National Bureau of Standards, Gaithersburg, MD.

Lopez, L., Elam, S., and Reed, K. (1989). "Software concept for checking engineering designs for conformance with codes and standards." *Engineering with Computers*, 5(2), 63-78.

Ma, H., Elsa, K., Chung, C., and Amor, R. (2006). "Testing semantic interoperability." *Proc., Joint International Conference on Computing and Decision Making in Civil and Building Engineering*, Montreal, Canada.

Ma, L., Sacks, R., Kattel, U., and Bloch, T. (2018). "3D object classification using geometric features and pairwise relationships." *Computer-Aided Civil and Infrastructure Engineering*, 33(2), 152-164.

Mahanta, J. (2017). "Introduction to neural networks, advantages and applications." <https://towardsdatascience.com/introduction-to-neural-networks-advantages-and-applications-96851bd1a207> (Nov. 18, 2018).

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." *Automation in Construction*, 32, 24-37.

Mandava, B., and Zhang, J. (2016). "A new automated quantity takeoff method for BIM-based bridge designs." *Proc., 33rd International Conference of CIB W78,* Brisbane, Australia.

Martins, J.P. and Monteiro, A. (2013). "LicA: a BIM based automated code-checking application for water distribution systems." *Automation in Construction*, 29, 12-23.

Marat, S., and Ltti, L. (2012). "Influence of the amount of context learned for improving object classification when simultaneously learning object and contextual cues: computational approaches to reading and scene perception." *Visual Cognition*, 20(4-5), 580-602.

Max, B. (2013). "Logic programming with Prolog." Springer, London.

Nadkarni, P.M., Ohno-Machado, L., Chapman, W.W. (2011). "Natural language processing: an introduction." *Journal of the American Medical Informatics Association*, 18(5), 544-551.

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

NSF. (2017). "EAGER/collaborative research: science-based exploration of invariant signatures of architecture/engineering/construction objects to enable interoperability of building info modeling." Award Abstract #1745374. <https://www.nsf.gov/awardsearch/showAward?AWD_ID=1745374&HistoricalAwards=false> (Jul 9, 2021).

Nawari, O. (2012). "Automating codes conformance." *Journal of Architectural Engineering*, 18(4), 315-323.

Nawari O. (2018). "Building information modeling: automated code checking and compliance processes." CRC Press, Boca Raton.

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

Nguyen, T., and Kim, J. (2011). “Building code compliance checking using BIM technology.” *Proc., 2011 Winter Simulation Conference (WSC),* IEEE, New York, 3395-3400.

Nelson, K. and Sokkappa, P. (2008). “A statistical model for generating a population of unclassified objects and radiation signatures spanning nuclear threats.” *International Nuclear Information System*, 40(17).

Paliouras, G., Spyropoulos, C.D., and Tsatsaronis, G. (2011). “Knowledge-driven multimedia information extraction and ontology evolution: bridging the semantic gap.” Springer, Berlin, Heidelberg.

Pazlar, T. and Turk, Z. (2008). “Interoperability in practice: geometric data exchance using the IFC standard.” *Journal of information technology in construction (ITcon),*13, 362-380.

Puente, I., González-Jorge, H., Martínez-Sánchez, J., and Arias. P. (2014). “Automatic detection of road tunnel luminaires using a mobile LiDAR system.” *Measurement: Journal of the International Measurement Confederation,* 47(1), 569-575.

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

Quintana, B., Prieto, S.A., Adán, A., and Bosché, F. (2018). “Door detection in 3D coloured point clouds of indoor environments.” *Automation in Construction*, 85, 146-166.

Quinlan, J.R. (1986). “Induction of Decision Trees.” *Machine Learning*, 1(1), 81-106.

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

Reid, P., Kelly, S., Dan, J., and Stefan, W. (2018). “Deconfounded lexicon induction for interpretable social science.” Proc., *2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 1, Association for Computational Linguistics, Stroudsburg, PA.

Ren, R., and Zhang, J. (2021). “Semantic rule-based construction procedural information extraction to guide jobsite sensing and monitoring.” *Journal of Computing in Civil Engineering*, accepted.

Rezac, J., Bim D., Gutten, O., and Rullisek, L. (2018). “Toward accurate conformational energies of smaller peptides and medium-sized macrocycles: MPCONF196 benchmark energy data set.” *Journal of chemical theory and computation*, 14(3), 1254-1266.

Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A., and Li, F. (2015). “ImageNet large scale visual recognition challenge.” *International Journal of Computer Vision,* 115(3), 211-252.

Rosandich, G. R. (1997). “Intelligent visual inspection using artificial neural networks.” Springer, New York, U.S.

Sacks, R. (2018). “BIM handbook: a guide to building information modeling for owners, designers, engineers, contractors and facility managers.” Hoboken, New Jersey Wiley.

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.” *Journal of Computing in Civil Engineering*, 31(6).

Sarawagi, S. (2008). “Information extraction.” *Foundations and Trends in Databases*, 3(3), 261-377.

Santos, A., Costa, A., and Grillo, A. (2017). “Bibliometric analysis and review of building information modeling literature published between 2005 and 2015.” *Automation in Construction,* 80, 118-136.

Seeliger, K., Güçlü, U., Ambrogioni, L., Güçlütürk, Y., and Gerven, M.A.J. (2018). “Generative adversarial networks for reconstructing natural images from brain activity.” *NeuroImage*, 181, 775-785.

Silver, D., Huang, A., Maddison, C.J., Guez, A., Sifre, L., van den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., Lanctot, M., Dieleman, S., Grewe, D., Nham, J., Kalchbrenner, N., Sutskever, I., Lillicrap, T., Leach, M., Kavukcuoglu, K., Graepel, T., and Hassabis, D. (2016). “Mastering the game of Go with deep neural networks and tree search.” *Nature*, 529(7587), 484.

Sionov, R.V., Vlahopoulos, S.A., and Granot, Z. (2015). “Regulation of BIM in health and disease.” *Oncotarget*, 6(27), 23058-23134.

Socher, R., Huval, B., Bhat, B., Manning, D.C., and Ng, Y.A. (2012). "Convolutional-recursive deep learning for 3D object classification." *NIPS'12: Proc., 25th International conference on neural information processing systems,* Springer, New York, NY, 656-664.

Solihin, W., and Eastman, C. (2015). "Classification of rules for automated BIM rule checking development." *Automation in Construction*, 53, 69-82.

Stow, D.A., Toure, S.I., Lippitt, C.D., Lippitt, C. L., and Lee, C. (2012). "Frequency distribution signatures and classification of within-object pixels." *International Journal of Applied Earth Observations and Geoinformation*, 15, 49-56.

STR Vision (2021). "Keep your job under control during planning and execution phases." < https://international.teamsystem.com/ww/str-vision-cpm> (Jul 20, 2021).

Spivey, J. M. 1996. "An introduction to logic programming through Prolog." London, New York, Prentice Hall.

Tan, X., Hammad, A., and Fazio, P. (2010) "Automated code compliance checking for building envelope design." *Journal of Computing in Civil Engineering*, 24(2), 203-211.

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

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

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

Veropoulos, K., Campbell, C., and Cristianini, N. (1999). "Controlling the sensitivity of support vector machines." *Proc.*, *International Joint Conference on AI,* 55-60.

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.

Volk, R., Stengel, J., and Schultmann, F. (2014). "Building information modeling (BIM) for existing buildings - literature review and future needs." *Automation in Construction*, 38, 109-127.

Wimalasuriya, D.C., and Dou, D. (2010). "Ontology-based information extraction: An introduction and a survey of current approaches." *Journal of Information Science*, 36(3), 306-323.

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. (2000). “Building extraction and reconstruction from Lidar data.” *International Archives of Photogrammetry and Remote Sensing,* 33(3), 958-964.

Whole Building Design Guide (WBDG). (2021). “Common building information model files and tools.” < https://www.wbdg.org/bim/cobie/common-bim-files> (Jul 20, 2021).

Witten, I.H., Frank, Eibe., Hall, M.A., and Paul, C.J. (2016). “Data mining.” Morgan Kaufmann, 4th Edition, Burlington, MA.

Won, J., Lee, G., and Cho, C. (2013). “No-schema algorithm for extracting a partial model from an IFC instance model.” *Journal of Computing in Civil Engineering*, 27(6), 585-592.

Wong Chong, O., and Zhang, J. (2021). “Logic representation and reasoning for automated BIM analysis to support automation in offsite construction.” *Automation in Construction*, 129(103756).

Wu, J., and Zhang, J. (2018a). “Automated BIM object classification to support BIM interoperability.” *Proc.*, *Construction Research Congress*, ASCE, Reston, VA, 706-715. DOI: 10.1061/9780784481301.070. With permission from ASCE.

Wu, J. and Zhang, J. (2018b). “Building information modelling (BIM) data repository with labels.” <https://purr.purdue.edu/projects/bimdatareposit/databases/> (Apr. 21, 2021).

Wu, J., and Zhang, J. (2019a). “Introducing geometric signatures of architecture, engineering, and construction objects and a new BIM dataset.” *Proc.*, *2019 ASCE International Conference on Computing in Civil Engineering*, ASCE, Reston, VA, 264-271. DOI: 10.1061/9780784482421.034. With permission from ASCE.

Wu, J., and Zhang, J. (2019b). “New automated BIM object classification method to support BIM interoperability.” *Journal of Computing in Civil Engineering*, 33(5). DOI: 10.1061/(ASCE)CP.1943-5487.0000858. With permission from ASCE.

Wu, J., Sadraddin, H.L., Ren, R., Zhang, J., and Shao, X. (2021a). "Invariant signatures of architecture, engineering, and construction objects to support BIM interoperability between architectural design and structural analysis." *Journal of Construction Engineering and Management,* 147(1). DOI: 10.1061/(ASCE)CO.1943-7862.0001943. With permission from ASCE.

Wu, J., Akanbi, T., and Zhang, J. (2021b). "Constructing invariant signatures for AEC objects to support BIM-based analysis automation through object classification." *Journal of Computing in Civil Engineering,* Submitted.

Wu, J., and Zhang, J. (2021a). "Model validation using invariant signatures and logic-based reasoning for automated building code compliance checking." *Journal of Computing in Civil Engineering,* Submitted.

Wu, J., and Zhang, J. (2021b). "Invariant signatures of architecture, engineering, and construction (AEC) objects in industry foundation classes (IFC)-based building information modeling." Purdue University Research Repository. doi:10.4231/7VW7-0129.

Xu, X., and Cai, H. (2020). "Semantic approach to compliance checking of underground utilities." *Automation in Construction*, 109, 103006.

Xue, X., and Zhang, J. (2020). "Building codes Part-of-Speech tagging performance improvement by error-driven transformational rules." *Journal of Computing in Civil Engineering*, 34(5), 04020035.

Xue, X., and Zhang, J. (2021). "Part-of-speech tagging of building codes empowered by deep learning and transformational rules." *Journal of Advanced Engineering Informatics*, 47(101235).

Xue, X., Wu, J., and Zhang, J. (2021). "Semi-Automated Generation of Logic Rules for Tabular Information in Building Codes to Support Automated Code Compliance Checking." *Journal of Computing in Civil Engineering*, in press.

Yang, Q.Z, and Xu, X. (2004). "Design knowledge modeling and software implementation for building code compliance checking." *Building and Environment*, 39(6), 689-698.

Yang, Z., Yu, H., Tang, J., and Liu, H. (2019). "Toward keyword extraction in constrained information retrieval in vehicle social network." *IEEE Transactions on Vehicular Technology*, 68(5), 4285-4294.

Yehia, E., Boshnak, E., AbdelGaber, S., Abdo, A., Elzanfaly, D.S. (2019). "Ontology-based clinical information extraction from physician's free-text notes." *Journal of Biomedical Informatics*, 98, 103276-103276.

Yin, Z., Li, R., Mei, Q., and Han, J. (2009). "Exploring social tagging graph for web object classification." *Proc., 15th ACM SIGKDD International Conference in Knowledge Discovery and Data Mining,* ACM, New York, 957-966.

Zeng, G., Song, R., Hu, X., Chen, Y., and Zhou, X. (2019). "Applying convolutional neural network for military object detection on embedded platform." *Communications in Computer and Information Science*, 994, 131-141.

Zhang, J., and El-Gohary, N. (2015). "Automated information transformation for automated regulatory compliance checking in construction." *Journal of Computing in Civil Engineering*, 29(4).

Zhang, J., and El-Gohary, N. (2016a). "Extending building information models semi-automatically using natural language processing techniques." *Journal of Computing in Civil Engineering, 30*(5).

Zhang, J., and El-Gohary, N. (2016b). "Semantic NLP-based information extraction from construction regulatory documents for automated compliance checking." *Journal of Computing in Civil Engineering*, ASCE, 1943-5487.

Zhang, J. and El-Gohary, N. (2016c). "Semantic-based logic representation and reasoning for automated regulatory compliance checking." *Journal of Computing in Civil Engineering,* 31(1), 4016037.

Zhang, J., and El-Gohary, N. (2017). "Integrating semantic NLP and logic reasoning into a unified system for fully-automated code checking." *Automation in Construction*, 73, 45-57.

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

Zhang, J., Fogelman-Soulié, F., Largeron, C. (2018). "Towards automatic complex feature engineering." *Lecture Notes in Computer Science*, 11234, 312-322.

Zhang, R., and El-Gohary, N. (2019). “A machine-learning approach for semantic matching of building codes and building information models (BIMs) for supporting automated code checking.” *Proc. GeoMEast 2019 International Congress and Exhibition*, Cairo, Egypt.

Zhou, N. (2014). “B-Prolog user’s manual (version 8.1): Prolog, agent, and constraint programming.” <http://www.picat-lang.org/bprolog/download/manual.pdf> (Apr. 1, 2021).

Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., and Torralba, A. (2017). “Scene parsing through ADE20K dataset.” *Proc.*, *of 30th IEEE Conference on Computer Vision and Pattern Recognition*, 2017, 5122-5130.

Zhou, B., Zhao, H., Puig, X., Xiao, T., Fidler, S., Barriuso, A., and Torralba, A. (2018). “Semantic understanding of scenes through ADE20k dataset.” *International Journal of Computer Vision*, 1-20.

Zou, Y., Kiviniemi, A., and Jones, S.W. (2017). “A review of risk management through BIM and BIM-related technologies.” *Safety Science*, 97, 88-98.

PUBLICATIONS

Wu, J., and Zhang, J. (2018a). “Automated BIM object classification to support BIM interoperability.” *Proc.*, *Construction Research Congress*, ASCE, Reston, VA, 706-715. DOI: 10.1061/9780784481301.070. With permission from ASCE.

Wu, J. and Zhang, J. (2018b). “Building Information Modelling (BIM) data repository with labels.” <https://purr.purdue.edu/projects/bimdatareposit/databases/> (Apr. 21, 2021).

Wu, J., and Zhang, J. (2019a). “Introducing geometric signatures of architecture, engineering, and construction objects and a new BIM dataset.” *Proc.*, *2019 ASCE International Conference on Computing in Civil Engineering*, ASCE, Reston, VA, 264-271. DOI: 10.1061/9780784482421.034. With permission from ASCE.

Wu, J., and Zhang, J. (2019b). “New automated BIM object classification method to support BIM interoperability.” *Journal of Computing in Civil Engineering*, 33(5). DOI: 10.1061/(ASCE)CP.1943-5487.0000858. With permission from ASCE.

Wu, J., Sadraddin, H.L., Ren, R., Zhang, J., and Shao, X. (2021a). “Invariant signatures of architecture, engineering, and construction objects to support BIM interoperability between architectural design and structural analysis.” *Journal of Construction Engineering and Management,* 147(1). DOI: 10.1061/(ASCE)CO.1943-7862.0001943. With permission from ASCE.

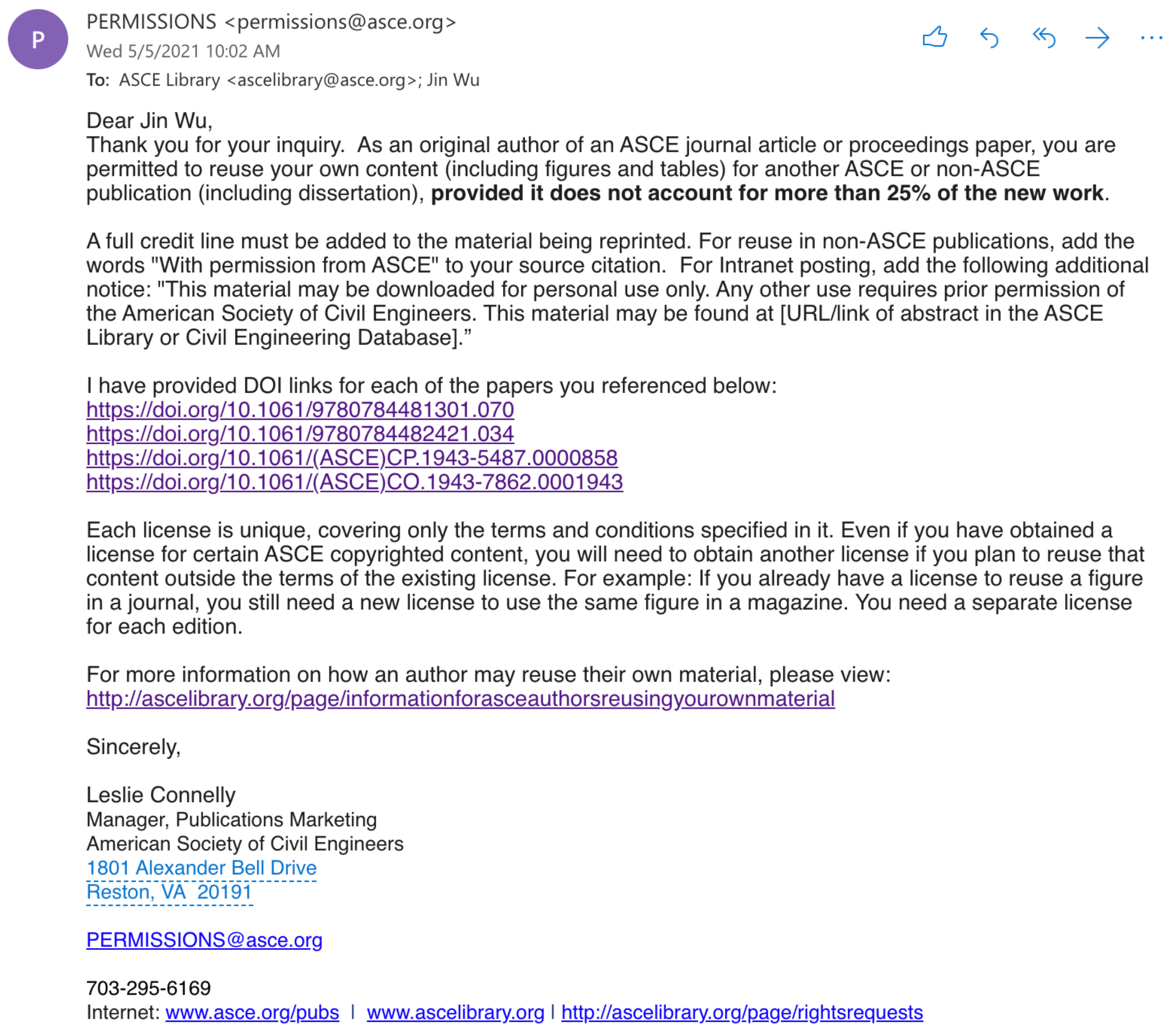
Wu, J., Akanbi, T., and Zhang, J. (2021b). “Constructing invariant signatures for AEC objects to support BIM-based analysis automation through object classification.” Submitted.

Wu, J., and Zhang, J. (2021a). “Model validation using invariant signatures and logic-based reasoning for automated building code compliance checking.” Submitted. With permission from ASCE.

Wu, J., and Zhang, J. (2021b). “Invariant signatures of architecture, engineering, and construction (AEC) objects in industry foundation classes (IFC)-based building information modeling.” Purdue University Research Repository. doi:10.4231/7VW7-0129.

APPENDIX A: MATERIAL REUSE PERMISSION FROM ASCE

Letter from the ASCE Publications Marketing Manager



PERMISSIONS <permissions@asce.org>
Wed 5/5/2021 10:02 AM
To: ASCE Library <ascelibrary@asce.org>; Jin Wu

Dear Jin Wu,
Thank you for your inquiry. As an original author of an ASCE journal article or proceedings paper, you are permitted to reuse your own content (including figures and tables) for another ASCE or non-ASCE publication (including dissertation), **provided it does not account for more than 25% of the new work**.

A full credit line must be added to the material being reprinted. For reuse in non-ASCE publications, add the words "With permission from ASCE" to your source citation. For Intranet posting, add the following additional notice: "This material may be downloaded for personal use only. Any other use requires prior permission of the American Society of Civil Engineers. This material may be found at [URL/link of abstract in the ASCE Library or Civil Engineering Database]."

I have provided DOI links for each of the papers you referenced below:
https://doi.org/10.1061/9780784481301.070
https://doi.org/10.1061/9780784482421.034
https://doi.org/10.1061/(ASCE)CP.1943-5487.0000858
https://doi.org/10.1061/(ASCE)CO.1943-7862.0001943

Each license is unique, covering only the terms and conditions specified in it. Even if you have obtained a license for certain ASCE copyrighted content, you will need to obtain another license if you plan to reuse that content outside the terms of the existing license. For example: If you already have a license to reuse a figure in a journal, you still need a new license to use the same figure in a magazine. You need a separate license for each edition.

For more information on how an author may reuse their own material, please view:
http://ascelibrary.org/page/informationforasceauthorsreusingyourownmaterial

Sincerely,

Leslie Connelly
Manager, Publications Marketing
American Society of Civil Engineers
1801 Alexander Bell Drive
Reston, VA 20191

PERMISSIONS@asce.org

703-295-6169
Internet: www.asce.org/pubs | www.ascelibrary.org | http://ascelibrary.org/page/rightsrequests

APPENDIX B: IMPLEMENTATION FIELDS OF INVARIANT SIGNATURES FOR CHAPTER 4

```java
// Object general fields
public String model;
public int stepline;
public String wholeIfcLine;
public String GUID;
public String name;
public String ifcName;
public String msg;
public boolean muiltObjs;
public int items, faces;
public boolean major_component;
public List<String> obj_msg = new ArrayList<>();
public List<InvariantSignature> sub_labels;
public String arg;
public String unit;
public boolean correct;

// Basic geometric information
public double length, width, height;
public double volume, area;

// Locational information
public Axis axis;
public String posH;
public String posV;
public int level;
public double mHigh, mLow, mRatio;
// placement x, y, default [0, 0] then ref diretion, default [1, 0], alt [0, 1]
public List<Double> axis_2d;

// Type
public String defaultType;
public int defaultTypeCount;

public String ruleType;
public int ruleTypeCount;
public String ruleTypeFull;

public String realType;
public int count;
```

```
// Regular = rectangular, circle, ring, I shapes
public boolean regular;

// Rectangular
public boolean Rec;
public double Rec_X, Rec_Y;

// Circle
public boolean Cir;
public double Cir_R;

// Ring
public boolean Ring;
public double R_R, R_T;

// I shape
public boolean I;
public double I_W, I_H, I_R, I_FT, I_WT;

// Extruded Area solid
public boolean Swept;
public double extruded_depth;
public Point extruded_direction_local;
public Point abs_extruded_direction;

// Brep
public boolean Brep;
public boolean valid;
public int[] vers = new int[100];
public double aveV;

// Surfact Model / Mapped represetation / Clipping / CSG
public boolean SurfaceModel;
public boolean Mapped;
public boolean Clipping;
public boolean CSG;
```

```
// Bounding Box V1
public double maxX, minX;
public double maxY, minY;
public double maxZ, minZ;

// Bounding Box V2 implentation within Axis
public double Xmax, Xmin, Ymax, Ymin, Zmax, Zmin;

// Local and global bounding box
public double x_low, x_high;
public double x_low_global, x_high_global;

public double y_low, y_high;
public double y_low_global, y_high_global;

public double z_low, z_high;
public double z_low_global, z_high_global;

// ToString helper
public StringBuilder invSigTag;
public StringBuilder invSigVals;
```

APPENDIX C: FULL STAAD OUTPUT OF THE REAL MODEL FOR SECTION 6.2

```
STAAD SPACE
START JOB INFORMATION
ENGINEER DATE 4-Aug-19
END JOB INFORMATION
INPUT WIDTH 54
UNIT FEET POUND
JOINT COORDINATES
1 20.13 -3.78 -3.95
2 20.13 0.0 -3.95
3 20.13 -3.78 3.65
4 20.13 0.0 3.65
5 20.13 -3.78 11.2
6 20.13 0.0 11.2
7 20.13 -3.78 18.75
8 20.13 0.0 18.75
9 20.13 -3.78 26.3
10 20.13 0.0 26.3
11 13.58 -3.78 -3.95
12 13.58 0.0 -3.95
13 13.58 -3.78 3.65
14 13.58 0.0 3.65
15 13.58 -3.78 11.2
16 13.58 0.0 11.2
17 13.58 -3.78 18.75
18 13.58 0.0 18.75
19 13.58 -3.78 26.3
20 13.58 0.0 26.3
21 6.98 -3.78 -3.95
22 6.98 0.0 -3.95
23 6.98 -3.78 3.65
24 6.98 0.0 3.65
25 6.98 -3.78 11.2
26 6.98 0.0 11.2
27 6.98 -3.78 18.75
28 6.98 0.0 18.75
29 6.98 -3.78 26.3
30 6.98 0.0 26.3
31 0.43 -3.78 -3.95
32 0.43 0.0 -3.95
33 0.43 -3.78 3.65
34 0.43 0.0 3.65
35 0.43 -3.78 11.2
36 0.43 0.0 11.2
37 0.43 -3.78 18.75
38 0.43 0.0 18.75
39 0.43 -3.78 26.3
40 0.43 0.0 26.3
41 20.18 4.68 -3.95
42 20.18 4.68 3.65
43 20.18 4.68 11.2
```

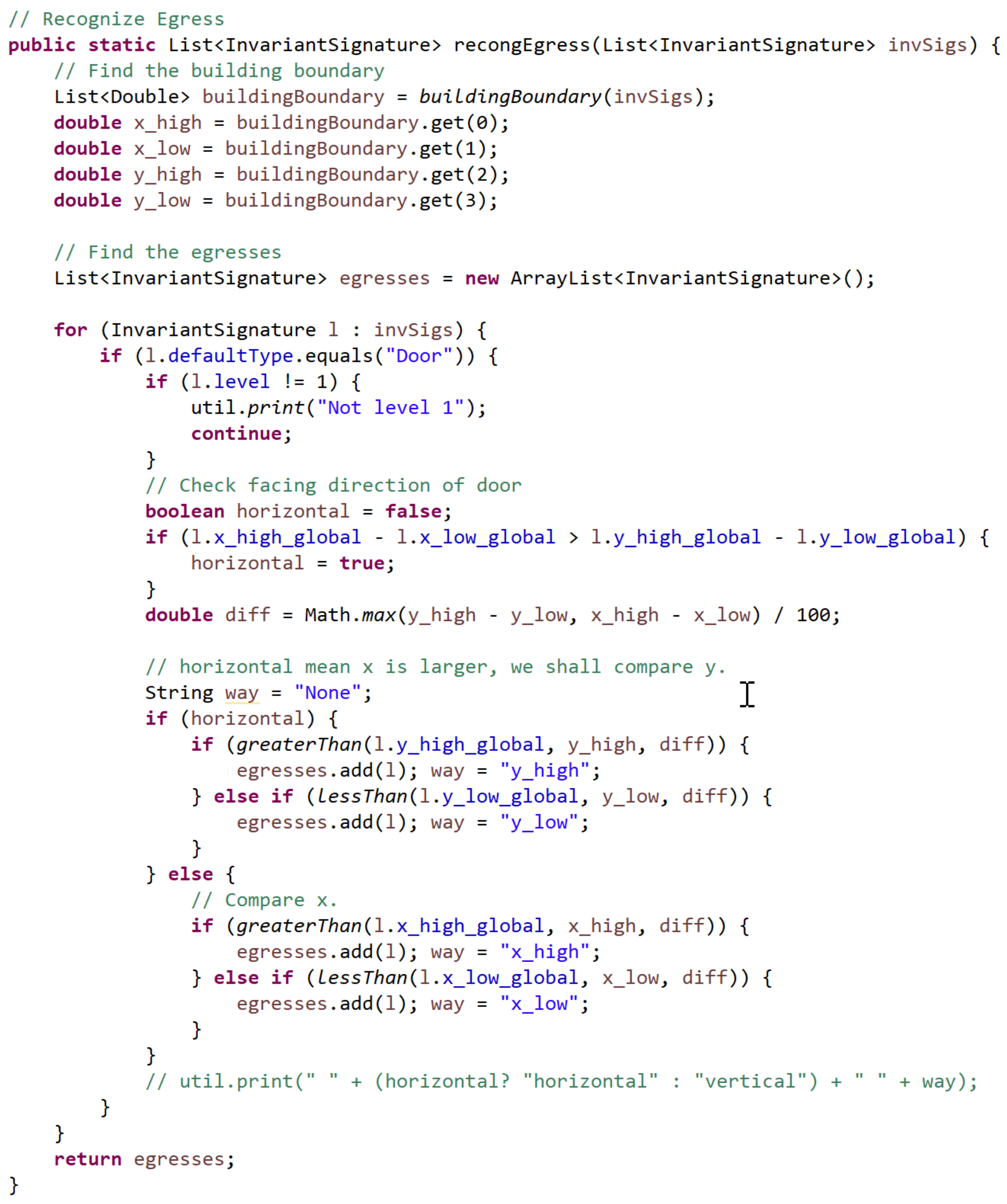
```
44 20.18 4.68 18.75
45 20.18 4.68 26.3
46 13.58 4.68 -3.95
47 13.58 4.68 3.65
48 13.58 4.68 11.2
49 13.58 4.68 18.75
50 13.58 4.68 26.3
51 6.98 4.68 -3.95
52 6.98 4.68 3.65
53 6.98 4.68 11.2
54 6.98 4.68 18.75
55 6.98 4.68 26.3
56 0.38 4.68 -3.95
57 0.38 4.68 3.65
58 0.38 4.68 11.2
59 0.38 4.68 18.75
60 0.38 4.68 26.3
61 6.98 7.68 18.75
62 6.98 7.68 26.3
63 0.38 7.68 18.75
64 0.38 7.68 26.3
MEMBER INCIDENCES
1 24 22;
2 34 32;
3 32 22;
4 34 36;
5 10 8;
6 8 6;
7 6 4;
8 4 2;
9 22 12;
10 12 2;
11 34 24;
12 24 26;
13 14 16;
14 36 38;
15 26 28;
16 16 18;
17 38 28;
18 28 18;
19 18 8;
20 40 30;
21 30 20;
22 20 10;
23 20 18;
24 14 12;
25 30 28;
26 40 38;
27 24 14;
28 14 4;
29 36 26;
30 26 16;
31 16 6;
32 50 49;
33 49 48;
34 48 47;
```

```
35 47 46;
36 55 54;
37 54 53;
38 53 52;
39 52 51;
40 57 52;
41 52 47;
42 47 42;
43 58 53;
44 53 48;
45 48 43;
46 59 54;
47 54 49;
48 49 44;
49 45 44;
50 44 43;
51 42 41;
52 43 42;
53 51 56;
54 41 46;
55 46 51;
56 59 60;
57 57 58;
58 58 59;
59 56 57;
60 50 45;
61 60 55;
62 55 50;
63 62 61;
64 63 61;
65 63 64;
66 64 62;
67 1 2;
68 3 4;
69 5 6;
70 7 8;
71 9 10;
72 11 12;
73 13 14;
74 15 16;
75 17 18;
76 19 20;
77 21 22;
78 23 24;
79 25 26;
80 27 28;
81 29 30;
82 31 32;
83 33 34;
84 35 36;
85 37 38;
86 39 40;
87 2 41;
88 4 42;
89 6 43;
90 8 44;
```

```
91 10 45;
92 12 46;
93 14 47;
94 16 48;
95 18 49;
96 20 50;
97 22 51;
98 24 52;
99 26 53;
100 28 54;
101 30 55;
102 32 56;
103 34 57;
104 36 58;
105 38 59;
106 40 60;
107 54 61;
108 55 62;
109 59 63;
110 60 64;
ELEMENT INCIDENCES SHELL
111 30 20 18 28
112 20 10 8 18
113 38 28 26 36
114 28 18 16 26
115 18 8 6 16
116 36 26 24 34
117 26 16 14 24
118 16 6 4 14
119 34 24 22 32
120 24 14 12 22
121 14 4 2 12
122 55 50 49 54
123 50 45 44 49
124 59 54 53 58
125 54 49 48 53
126 49 44 43 48
127 58 53 52 57
128 53 48 47 52
129 48 43 42 47
130 57 52 51 56
131 52 47 46 51
132 47 42 41 46
133 64 62 61 63
ELEMENT PROPERTY
111 TO 133 THICKNESS 0.2
DEFINE MATERIAL START
ISOTROPIC CONCRETE
E 4.536e+008
POISSON 0.17
DENSITY 149.99
ALPHA 5.5e-006
DAMP 0.05
TYPE CONCRETE
STRENGTH FCU 576000
END DEFINE MATERIAL
```

```
CONSTANTS
MATERIAL CONCRETE ALL
MEMBER PROPERTY AMERICAN
1 TO 66 PRIS YD 0.4 ZD 0.5
67 TO 110 PRIS YD 0.5 ZD 0.5
FINISH
```

APPENDIX D: EGRESS RECOGNITION ALGORITHM FOR CHAPTER 7



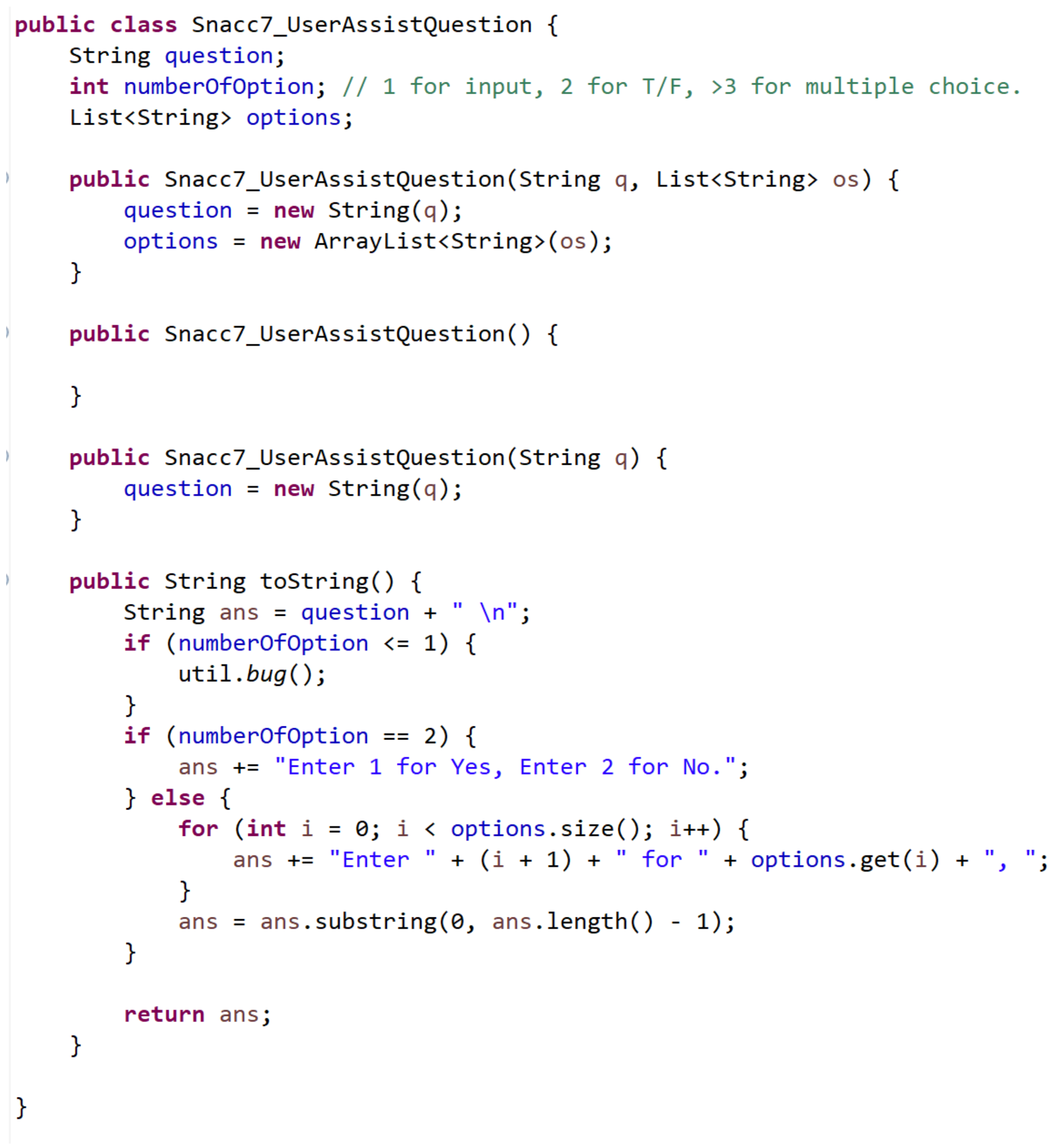
```java
// Recognize Egress
public static List<InvariantSignature> recongEgress(List<InvariantSignature> invSigs) {
    // Find the building boundary
    List<Double> buildingBoundary = buildingBoundary(invSigs);
    double x_high = buildingBoundary.get(0);
    double x_low = buildingBoundary.get(1);
    double y_high = buildingBoundary.get(2);
    double y_low = buildingBoundary.get(3);

    // Find the egresses
    List<InvariantSignature> egresses = new ArrayList<InvariantSignature>();

    for (InvariantSignature l : invSigs) {
        if (l.defaultType.equals("Door")) {
            if (l.level != 1) {
                util.print("Not level 1");
                continue;
            }
            // Check facing direction of door
            boolean horizontal = false;
            if (l.x_high_global - l.x_low_global > l.y_high_global - l.y_low_global) {
                horizontal = true;
            }
            double diff = Math.max(y_high - y_low, x_high - x_low) / 100;

            // horizontal mean x is larger, we shall compare y.
            String way = "None";
            if (horizontal) {
                if (greaterThan(l.y_high_global, y_high, diff)) {
                    egresses.add(l); way = "y_high";
                } else if (lessThan(l.y_low_global, y_low, diff)) {
                    egresses.add(l); way = "y_low";
                }
            } else {
                // Compare x.
                if (greaterThan(l.x_high_global, x_high, diff)) {
                    egresses.add(l); way = "x_high";
                } else if (lessThan(l.x_low_global, x_low, diff)) {
                    egresses.add(l); way = "x_low";
                }
            }
            // util.print(" " + (horizontal? "horizontal" : "vertical") + " " + way);
        }
    }
    return egresses;
}
```

APPENDIX E: USER ASSISTED CONCEPT’S QUESTION DATA STRUCTURE FOR CHAPTER 7



```java
public class Snacc7_UserAssistQuestion {
    String question;
    int numberOfOption; // 1 for input, 2 for T/F, >3 for multiple choice.
    List<String> options;

    public Snacc7_UserAssistQuestion(String q, List<String> os) {
        question = new String(q);
        options = new ArrayList<String>(os);
    }

    public Snacc7_UserAssistQuestion() {

    }

    public Snacc7_UserAssistQuestion(String q) {
        question = new String(q);
    }

    public String toString() {
        String ans = question + " \n";
        if (numberOfOption <= 1) {
            util.bug();
        }
        if (numberOfOption == 2) {
            ans += "Enter 1 for Yes, Enter 2 for No.";
        } else {
            for (int i = 0; i < options.size(); i++) {
                ans += "Enter " + (i + 1) + " for " + options.get(i) + ", ";
            }
            ans = ans.substring(0, ans.length() - 1);
        }

        return ans;
    }

}
```