Identifying Unlabeled 3D Components in BIM Models for Industrial Projects

by

Sina Abdollahnejad

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

in

Construction Engineering and Management

Department of Civil and Environmental Engineering

University of Alberta

Abstract

Oil and gas projects are known for their size and complexity and incorporate multiple disciplines such as concrete, steel, and piping. Each discipline is executed within a confined area during a limited timeframe. The execution of each discipline requires careful planning and coordination between the different disciplines. Each discipline creates separate Building Information Models (BIM), which are merged together with others into one huge model in terms of complexity. This model is used for different purposes such as: coordinating work packages and detecting any possible clashes.

From the contractor perspective, the BIM model can be utilized for defining the scope and obtaining a preliminary estimate while the project is in its early stages. The model's value depends on its degree of completeness and time that it will be available. However, lack of standard structure for Building Information Modeling in the industry causes immature, inconsistent and incomplete BIM models during the early stages of the project. This means the model usefulness becomes limited, thus the contractor has to review the model manually to extract the required and useful information, including the scope of each discipline, a preliminary estimate of quantities, etc.

The objective of this research is to investigate and develop a new methodology that can automatically fill the missing data in the BIM model and leverage its usage. This objective is achieved by identifying the type for each BIM model component using convolutional neural networks. This approach focuses on different projections of BIM model components rather than incomplete descriptive attributes. The research reviews different 3D image classification methods to select the most suitable method. After selecting a suitable method, an image classifier is

developed to identify the missing labels for the BIM model components. Then, the methodology is validated by using three real-world industrial project models. Results indicate that the proposed method can automatically process ill-defined and incomplete BIM models to fill the missing data, and works with 91% accuracy on classifying the BIM model components.

Acknowledgements

I would like to express my gratitude and appreciation to my supervisor Dr. Yasser Mohamed for his kind advice, continuous support and encouragement.

I also wish to thank Rick Hermann and Dr. Di Hu for providing me with the opportunity to do this project work in PCL Industrial Management Inc.

I am grateful to my wife and my parents, who have provided me with moral and emotional support in my life.

Contents

List of Tables

List of Figures

1. Introduction

Industrial projects were among the first to use building information modeling technology; this is mainly due to the project's complexity, which have increased exponentially in the last ten years [1]. Studies have shown that complexity of a project determine the utilization degree of information technology in a project [2]. Since industrial projects include many component types and require a careful plan and coordination, they use modelling tools, such as Navisworks®, which can merge multiple 3D models. Because of confusion around the definition of BIM, there is no consensus about whether or not these types of software should be considered BIM tools [3].

Building Information Modelling (BIM) has become a modelling standard in construction industry; it provides a virtual environment for the Architecture, Engineering, and Construction (AEC) industry where they can generate, exchange, and merge to increase collaboration and productivity in the projects. Many case studies have shown the benefits of BIM in construction projects [3]–[5]. These benefits are observed by both the private companies and government agencies all around the world, which have started to mandate BIM models for public sector construction projects [6], [7]. BIM can be used in two categories of passive and active. Passive usage of BIM encompasses engineering analysis like safety and scheduling. On the other hand, active usage involves extracting embedded knowledge in BIM [8]. BIM has excellent potential, but there are some unsolved issues in its modelling process. These issues can be grouped into two categories of contractual and technical issues [4].

Some researchers argue that a model that only contains 3D objects like CAD models and not having other information, or with having few attributes such as type, material, etc. should not be considered as a BIM model [3]. According to the definition, industrial models in the early stages

of a project, in which the models lack sufficient attributes, are merely 3D models. Other researchers divided BIM tools to the tools that are capable of handling 3D objects' classes and relationships and the ones that do not provide full BIM capabilities such as Autodesk Navisworks [9]. The third group of researchers consider models with few attributes as BIM models [10], [11]. In this research, the generic definition of BIM has been used, which includes models containing 3D objects with few attributes.

Typically, an industrial project BIM model consists of multiple sub-models such as structural, mechanical, electrical, etc. Models are designed in parallel to each other, and then all models are compiled by an engineering firm into one model to be reviewed. After that, the engineering firm issues the model to the contractor as one model. Figure 1 shows an IDEF0 diagram for this process. This process is repeated multiple times for fast-tracked projects. IDEF0 diagram is a method for diagramming processes, in which each box represents a process, and arrows on the left, top, right, and bottom are inputs, controls, outputs and mechanisms, respectively.

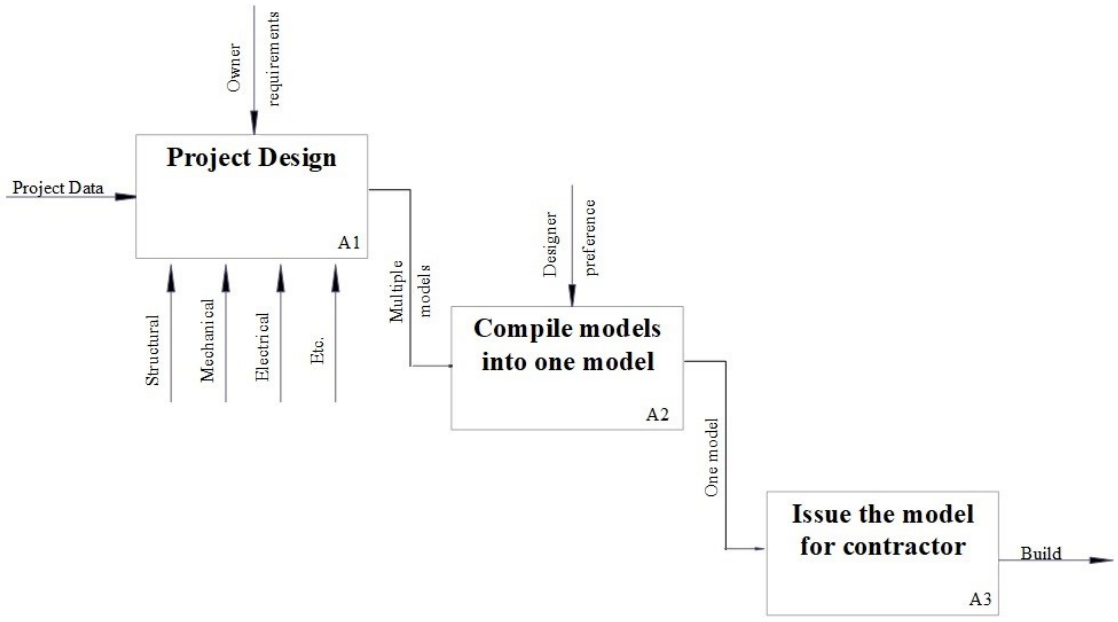


Figure 1. IDEF0 for issuing a BIM model to the contractor.

The above process results in the following concerns in fast-tracked industrial projects:

1. BIM ownership (contractual level): The contractor receives only the compiled model for reviewing and visualization. However, the contractor is unable to modify the model, or add other attributes like component type to the model and, therefore, the contractor has to save all operational attributes in a separate database.
2. Lack of standards (contractual level): The same item might be labelled as "I beam," "I-Beam," or "I Beam column" according to the engineering firm's conventions as there is no common standard.
3. Model limitation (technical level): The contractor cannot calculate quantity take-off accurately as the component types are not accurate.
4. Interoperability (technical level): Transferring data between different systems or software is a time-consuming, and error-prone task. For instance, receiving software might drop unsupported classes and properties [12].

During a project's life cycle, different parties generate a huge number of documents, CAD drawings, and BIM models [13]. Integrating these heterogeneous data from differeny sources involves utilizing essential applications such as 4D visualization software [14], and merging BIM and GIS data [15], etc. Since these data come in different formats that are optimized for a specific application, the machines cannot automatically process and link them, and human inputs are required to link the data between different sources. In addition, the BIM models are not shared properly in industrial construction projects due to issues such as liability, which results in the models that are not complete and called dump models.

In summary, there are two limitations that data usage in fast-tracked industrial projects suffers from. First, the lack of data integrity and completeness in the BIM models, which limits the model usability for preliminary analysis. And Second, although the construction industry is information intensive [16], there is no common standard for transferring and merging heterogeneous data between different data sources, and these two tasks are usually performed on an ad-hoc basis.

1.1. Research Objective

The objective of this study is based on the following hypothesis: During the early stages of a fast-tracked industrial project, automated solutions can be developed to fill in missing data in BIM models.

More specifically, this research aims to provide an automated solution to leverage data usage in BIM models at early project stages to complete and validate inconsistent and missing data in the BIM models, focusing mainly on identifying the type of BIM model components based on their geometry using convolutional neural network algorithm. It aims to investigate whether a CNN architecture can provide accurate classification of those components and what level of accuracy can be achieved.

1.2. Research Methodology

The research methodology focuses on identifying the component types in BIM models. Different 3D image classification methods, found in the literature, are reviewed and analyzed. Based on this

analysis, a 3D image classification method [17] is selected. This technique is modified and adapted for the problem's domain and then used to label the unidentified BIM components.

The proposed methodology is validated using three real-case oil and gas projects that have been executed in Alberta. Each step of the methodology is validated as follows:

The 3D image classification algorithm is applied to the components in three projects. Then a sample is drawn from the projects to be identified manually to calculate the accuracy. Figure 2 shows the different stages of the present research.

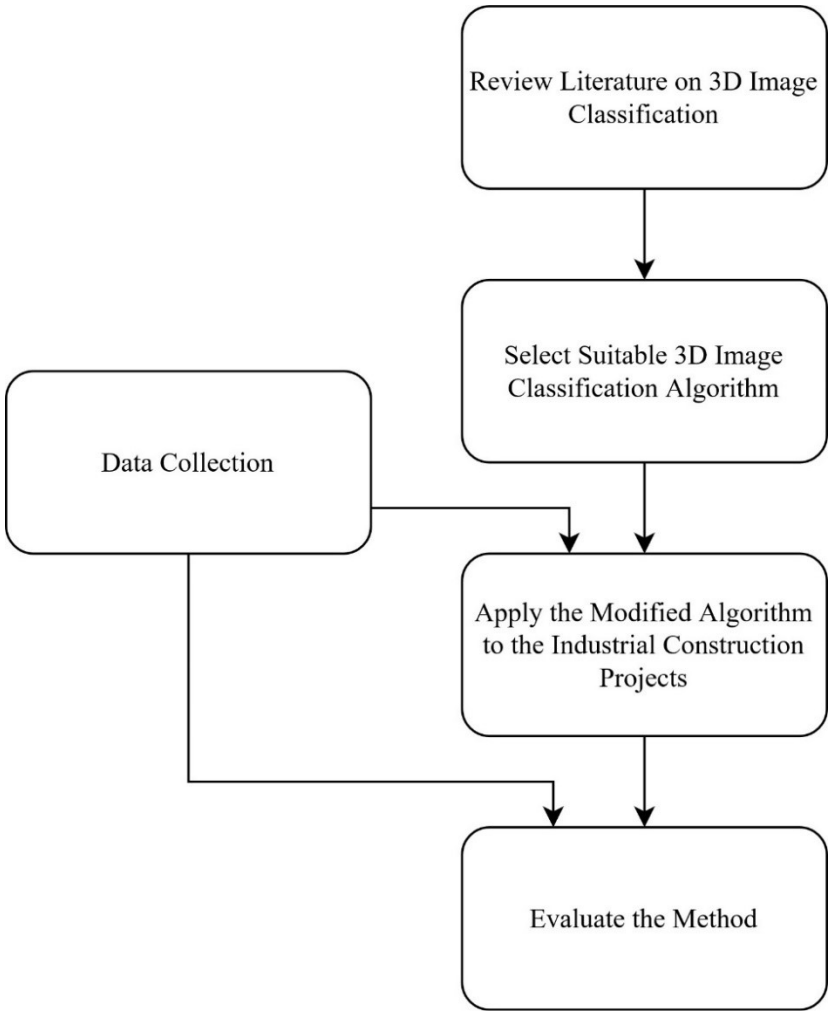


Figure 2. Research methodology flowchart.

1.3. Thesis Organization

The remainder of the thesis is organized as follows. Chapter 2 reviews the literature and different 3D image classification methods. Chapter 3 discusses the use of 3D image classification methods to identify the label for BIM model components. It starts by reviewing artificial neural networks and convolutional neural networks. Afterwards, it elaborates on the data preparation step and the proposed methodology. Chapter 4 discusses the implementation of the proposed method, and in chapter 5, the model is validated.

Finally, Chapter 6 concludes this research and summarizes the academic and industrial contributions, limitations, and future work. Appendix A, Appendix B, Appendix C, and Appendix D show the sample MATLAB code for dataset generation, Developed CNN model, Naviswork Plugin code, and viewpoint generation code, respectively.

2. Literature Review

2.1 BIM in Industrial Projects

Industrial projects are larger and more complex than the building projects and include information and design technology more than other types of construction projects [2]. Although industrial construction projects have many similarities with other types of construction projects, they also have some specific characteristics. Because of their final product, this type of construction projects is known for being more complex and utilizing more sophisticated management tools [18].

In order to reach the market faster and gain a faster return on investments, industrial projects are executed utilizing fast-track contracts [19]. In this type of contract, design and build stages are performed in parallel to each other, instead of being performed sequentially. In other words, the construction stage usually starts before the design is finalized. This requires the contractor to consider the new designs in the construction plans, as there are usually changes in the design as the project progresses [20].

Industrial construction projects are known for being complex because of several factors. First, complexity in industrial projects is due to lack of clearly defined scope at the start of the project. The scope of an industrial project is usually defined through a procedure named of Front End Loading (FEL) planning. FEL procedure also include the constructability and maintainability reviews [18].

The second source of complexity is that there are higher degrees of managerial and technical risks in industrial projects. Managerial risks may include schedule delays, scope creep, budget overruns,

etc. And the technical risks in engineering, procurement or construction include explosions, leak of extremely hazardous materials or severe environmental damages [18].

The third source of complexity is that due to the complicated nature of industrial projects, they require significant amounts of coordination and sophisticated project management. In contrast to residential construction project, an industrial project starts as a request from an owner to build, modify or demolish a plant. Then the owner hires an engineering only, Engineering, Procurement and Construction Management or Engineering, Procurement and Construction firm(s) to perform the FEL planning and defines the scope of the project [18].

Many industrial projects utilize modular construction as a part of their project execution plans. Modular construction relies on large volumetric components or as substantial elements of a building that are factory-produced pre-engineered building units (modules) that are delivered to the site and assembled. As the modules are usually manufactured off-site earlier and are shipped to the site for final installation, a prefabrication paradigm is utilized in fast-track projects [19], [20].

There are multiple processes in industrial construction projects that are further explained below. Nevertheless, some of these processes are done in parallel to the others in industrial construction projects in fast-track environment.

The first stage of the industrial projects are the feasibility study process. Any industrial project starts as a project idea that needs to be evaluated. These ideas are either stay-in-business or revenue generating projects. Stay-in-business projects are mandatory in order to comply with external regulations and have to be performed regardless of their cost or expected revenue. Therefore, these projects directly go to the planning stage. On the other hand, revenue generating projects have to

be studied for their feasibility. The major outcome of feasibility study is to make sure that the expected revenue meets the threshold set by the owner(s). After the owner(s) decide to proceede with project based on the feasibility study, the planning stage starts [18].

In the engineering phase, the engineers receive contracts from the owner(s), and they initiate internal projects within their organizations providing them with the required services. The project has to initiate with a well-defiend scope and accurate engineering documents to assure the success of the project. For industrial projects, these are developed during the planning and engineering phase through Front End Loading (FEL) planning. FEL planning usually consists of three processes named FEL I, FEL II, and FEL III [18].

In FEL I, all the alternative solutions to accomplish the desired product are identified. In FEL II, the best solution is identified after evaluating all the alternatives identified in the previous process. In the FEL III process, the selected alternative is developed to a complete set of design specifications. The rest of the processes in this phase include the detailed engineering and design, shop drawings, procurement and construction support, and as-builting processes [18].

The next phase is the procurement phase. In industrial projects, procurement is mostly handled by the procurement divisions in an Engineering Procurement and Construction (EPC) firm, which acts on behalf of the the construction firms, or owner(s). This arrangement requires a tremendous amount of efforts, integration and interface management. Some of the processes in the procurement phase include the engineering support, requisition, bidding and awarding, contract administration, and materials management processes [18].

The last phase is the construction phase, and include several processes such as: engineering support, fabrication, assembly, and site installation process. In the engineering support process

the constructors provide their input to the engineers to help them with the design optimization, site layout, and etc. In the fabrication process, the products get fabricated according to the shop drawings and get shipped to the assembly yards or construction site. In the assembly process, the fabricated parts get assembled in module yards outside of the construction location. And finally in the site installation process, activities like site preparation, rough and final grading, pilling, foundations, and modules installation, etc. take place [18].

Although there are several challenges during different phases of industrial projects, using BIM can help satisfy different aspects of utilizing fast-track technique in industrial projects, as they require careful planning and coordination. Building Information Modeling (BIM) represents the physical and functional characteristics of a facility in a digital format. Since industrial projects consist of many disciplines and require careful planning and coordination, BIM has been applied widely in industrial construction projects, such as Autodesk Navisworks, Intergraph SmartPlant and Bentley products. The BIM models provide a platform to create, manage and share 3D digital representations for any construction projects such as commercial, industrial, and transportation projects. In the BIM models, each object contains some information about it. This information includes properties of accurate geometrical representation, as well as non-geometric properties, such as structural properties, cost of materials, information of assembly, life cycle cost and environmental data [21]. The information in the BIM models makes the models widely applicable and can be used for multiple purposes. Since the base of the BIM models is 3D computer graphics modelling, they are useful for viewing, demonstration and 3D rendering.

To address the limitations of using the BIM model in quantity take-off, the absent information about the model components need to be extracted. In the quantity take-off process, one of the primary information required is the type of each model component.

Information about the type of each model component can be obtained by different methods. As shown in Figure 3, the 3D viewing model retains the properties and descriptions from the original BIM model. The information about the component type can be accessed in the text data format from the properties and description section for each component. However, as there are several engineering firms present in large construction projects such as industrial projects, different engineering firms have their internal standards and naming rules. Therefore, it is hard to group components in BIM models using semantic information because of different conventions between model designers and model users and lack of standards [22], [23]. For example, the value of the "TYPE" property could be "SCTN," "I-SCTN," and "BEAM," and all these values can be indicated as the component type of "I-Beam." In addition, even for two components that both have the value of "TYPE" as "Beam," it is not always clear that the components are steel beams or concrete beams.

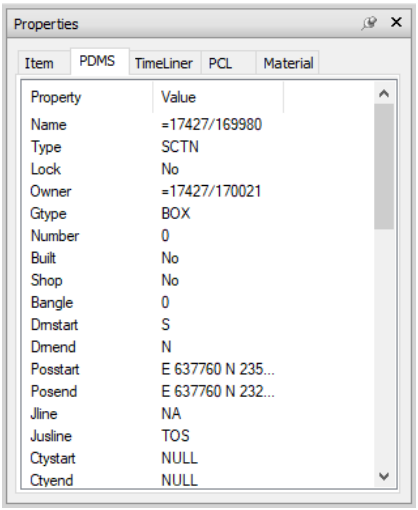


Figure 3. The properties of the model component in Autodesk Navisworks.

The text-based grouping recognition method was built to obtain information about the types of model components. The method sets some filters as a series of rules to group 3D model

components based on the properties of the components that have been extracted. These properties are stored in a separate database as text data. However, the filters need to be custom-built for each project, and it is not capable of recognizing all model components with the component types in the quantity take-off table [24].

Another attempt to obtain information about types of components from the BIM models has been based on the shape of the model component. The geometrical shape of the same type of components is the same but on different scales. On the other hand, for the components that have different types with the same geometrical shape, the geometrical features required for the quantity take-off process are the same. In other words, the shape of 3D model components is important in the quantity take-off process. Several general algorithms have been built to recognize and retrieve the shape of 3D model components [24]. Computer vision and object recognition have also been used to classify the components in the 3D models, especially for models based on cloud point data or feature point vector data. In such models, the data can be converted to solid models [25] and recognized for quantity take-off. Specifically, for industrial construction areas, the shape of the components in the 3D model can be grouped and automatically classified based on 3D mesh data [26].

In the shape recognition algorithm proposed by Ali[26], the shape descriptor is used to represent different shapes in the 3D models. The shape descriptor is defined as the histogram. These histograms are constructed from different shape functions. In the triangle-mesh-based 3D model, the shape functions are formed based on the vertex points of the triangle mesh. By comparing the shape descriptor histogram of each component using pre-defined dissimilarity measures, it is able to classify the 3D model components. Although the shape recognition algorithm proposed by Ali

is able to recognize the shape in industrial projects with an 82% success rate, it can only be utilized in recognizing the steel discipline.

A similar technique is used to obtain as-built models using laser scanning. According to Tang et. al. [27], creating an as-built model using laser scanning technology requires three main steps: 1) Data collection: surveying techniques will obtain a dense cloud of points that accurately measures the physical facility; 2) Data preprocessing: as multiple laser scanners must be used to capture different faces of the facility, the collected points must be registered in a single coordinate system; and 3) Modelling in BIM: different objects should be identified and categorized in the BIM model using the collected point cloud.

In detecting objects from the point cloud process, the steps include identifying a 3D object (typically a physical object in the point cloud but a digitized virtual object in our case) by collecting points from the objects' surfaces. Then, the data has to be preprocessed by being cleaned, smoothed, and having its outliers removed. Afterwards, a surface model can be generated using a curve-net-based method or by a polygon-based modelling method [28]. Finally, these surfaces can be used to construct the 3D surface object [29]–[32]. Although many algorithms try to construct 3D objects from point clouds [33]–[36], using point clouds to model different objects in BIM is a manual task that consumes most of the time required to create an as-built model [27], [34].

2.2. 3D Object Classification

Since the late 1970s, Computer-Aided Design (CAD) has replaced traditional paper drawing [37], because of better quality, quick and accurate editing, and productivity increase. Since then, CAD has revolutionized the design process in both engineering and academia [38], [39].

One of the limitations of CAD systems is the lack of objects' attributes concept, which constraints their ability to share data between different systems [40]. Therefore, BIM quickly superseded CAD systems as BIM seems to address CAD's limitations, such as attributes of objects. BIM provides a huge information source with search and analysis capabilities [37].

In recent years, the rapid increase in the number of 3D objects has required new methods to search and retrieve these objects as a traditional text search is insufficient [41]. There are many studies regarding 3D object classification [24], [42], [43] that have been mostly done in the computer science field.

The 3D object classification process can be divided into two parts, namely data representation of 3D objects and training of the classifier on the represented data. The 3D objects can be represented by two main methods, namely, shape descriptors and convolutional neural networks [44].

Shape Descriptors: A large number of shape descriptors have been developed in the computer vision field. For instance, shapes can be represented as histograms or bag-of-feature models that are constructed from surface curvatures [45]. Alternatives shape descriptors include models based on distances, angles, or triangle areas [46], local shape diameters measured at densely sampled surface points [47]. Heat kernel signatures [48], [49], or extensions of Scale-Invariant Feature Transform (SIFT) and Speeded-Up Robust Features (SURF) feature descriptors to 3D voxel grid

[50]. The spherical harmonic descriptor (SPH) [51] and the Light Field descriptor (LFD) [52] are other popular descriptors. LFD extracts geometric, and Fourier descriptors from object silhouettes rendered from several different viewpoints and can be directly applied to the object classification task. Since these features are hand-crafted, some of them do not generalize well across different domains [53].

Convolutional Neural Networks: Convolutional Neural Networks (CNNs) [54] have been successfully used in different areas of computer vision and beyond. In particular, significant progress has been made in the context of learning features. It turns out that training from large RGB image datasets (e.g., ImageNet [55]) is able to learn general-purpose image descriptors that outperform handcrafted features for several vision tasks, including object detection, scene recognition, texture recognition and classification [56]. This significant improvement in performance on these tasks has decidedly moved the field forward.

The data representation approaches can be coarsely classified into three categories according to their input: 1) view-based methods, 2) volume-based methods, and 3) point cloud-based methods. Among the three categories, the view-based methods generally outperform the other two.

View-Based methods: View-based methods have been considered as one of the fundamental approaches in 3D object classification. View-based methods project 3D objects into multiple 2D views; then, the classification is conducted using the features from 2D CNNs. Figure 4 shows an example of input data of a channel section in steel structure for the view-based CNN model.

Figure 4. An example of view-based 3D object.

Volume-based methods: Volume-based approaches apply a 3D convolutional neural network directly on voxelized shapes. Voxel-based methods learn 3D features from voxels, which represent 3D shape by the distribution of corresponding binary variables. Figure 5 shows an example of a voxelized 3D object.

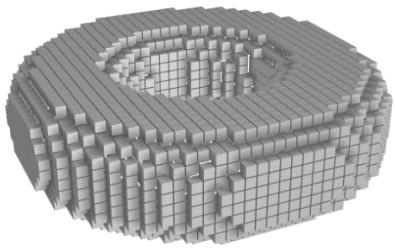


Figure 5. An example of a voxelized 3D object.

Point cloud-based methods: While previous works often combine hand-crafted features or descriptors with a machine learning classifier, the point cloud-based methods operate directly on point clouds in an end-to-end manner. Figure 6 shows a constructed point cloud of a 3D object.

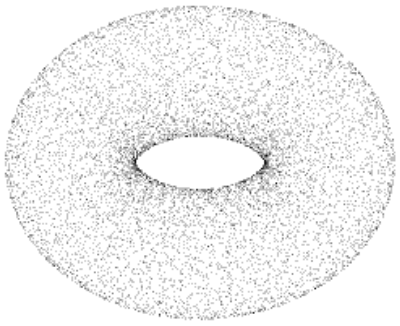


Figure 6. An example of a point cloud of a 3D object.

In similar research, shape descriptors to identify the component types in industrial BIM models [26]. However, the convolutional neural network has never been used in this field of the construction industry, in spite of having promising results in other fields like computer vision. In addition, the proposed methodology was only able to identify the steel structure, and the developed algorithm was not tested on piping components of the project [26].

Convolutional neural networks have been utilized in construction industry in several applications. In a research, CNN was used to detect concrete carcks and evaluate its density and reached approximately 90% for both the max F1 and AP scores on training, validation, and test sets [57]. In another research, the problem of the building quality problem in the Chinese government using convolutional neural networks [58]. In addition, there have been several researches about utilizing transfer learning technique based on the standard datasets that have been generated for object classification purposes [59].

In summary, there are several ways to represent the data to a object classification operator shape descriptors and convolutional neural networks. According to the literature review, since features

created by the shape descritors are hand-crafted, some of them do not generalize well across different domains [53], and the promising performance of the CNN in the field of 3D object classification, it was decided to opt the CNN for this research. In convolutional neural networks, the data can be represented in three different ways named view-based, voxel-based, and point cloud-based. In this research, a view-based convolutional neural network is opted to be used to identify the labels for BIM model components, because of its promising capability in the classification of 3D objects and outperforming other two methods because of its better generalizability [59] and the simplicity of its architecture, as the input data are simple images.

3. Framework Design and Development

The present research proposes an alternative method to classify and extract the “TYPE” attribute of the 3D model components for quantity take-off.

After reviewing different methods, a CNN model is opted to be used due to its performance in image classification tasks. In the present research, there are two sources of building the dataset. The first one is the artificially generated images that were generated using MATLAB. These projections are the inputs for the CNN model. After training the model using the projections, the model is saved using transfer learning to be used in a different environment. In order to utilize the pre-trained model, the model has to be retuned using the actual images that are generated from the second source which is the 3D model. In this stage, only a few images are needed for retuning, unlike the main training of CNN that a bigger dataset is required. After that, the model is tested using the images generated from the 3D model that were not shown to the model before.

The processing includes three major parts: The selected components will be isolated from the 3D BIM model. Then four viewpoints in four projections will be generated for each 3D components using a developed application program interface (API). After the viewpoints are merged in a table and exported as an image file, the CNN algorithm will be applied to each model component to obtain the “TYPE” attribute for them. The output of the method is the components’ type, which will be saved in the database, which can provide information for the quantity take-off process. Figure 7 shows the overall research framework.

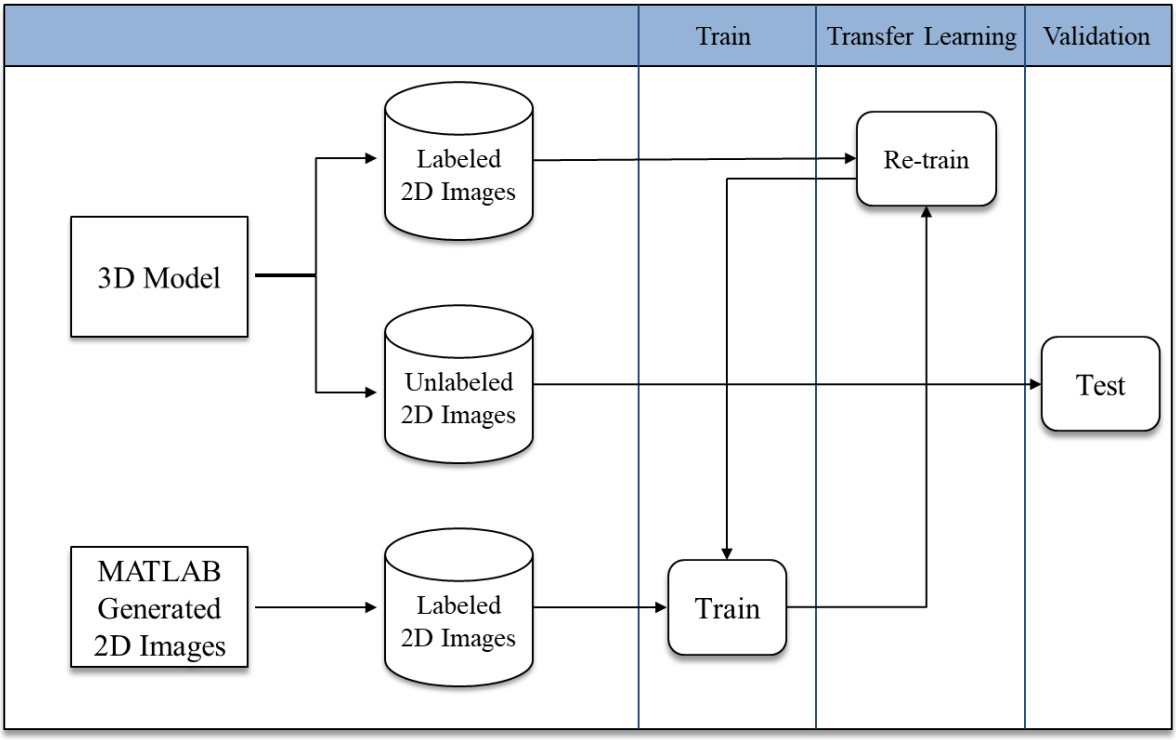


Figure 7. The overall framework of the research.

3.2. Data Preparation

In order to develop the machine learning model, a set of data is required to train the model based on them. To build a dataset, first, the characteristics of suitable data for the model should be identified. Since view-based CNN will be used in the research, the required data should include different projections of each 3D model component.

As the components in industrial BIM models usually have simple shapes (e.g. I-Beams, tubes, angles), only four projections will be included in each training record. These projections for each 3D component are front, back, side and top view, which are all merged into one image file. For example, Figure 8 shows a 3D component in the BIM model, which is an isolated steel column. Figure 9 shows the image that is suitable for training the multi-view CNN, as it consists of different projections of the 3D component shown in Figure 8.

Figure 8. An isolated 3D component in the BIM model.

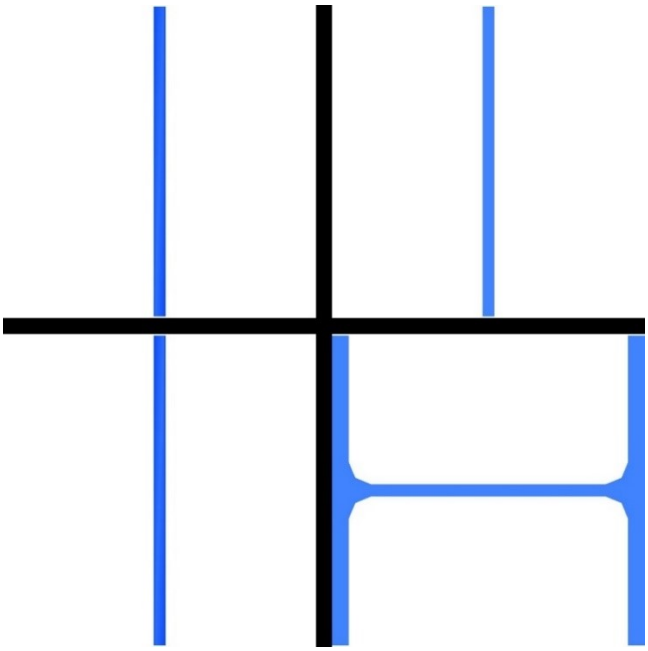


Figure 9. An image generated by Autodesk Navisworks plugin containing projections.

There are several disciplines in industrial BIM models, and two of the most important ones in the quantity take-off process are steel and piping disciplines. The mentioned disciplines might contain thousands of components in industrial projects. Each discipline in industrial BIM models contains several component types, some of which are more important than others, as they have more substantial quantities and are more expensive than the other components. For example, beams and columns are more important than the base plates in steel structure, as, in industrial models, there are more beams and columns than base plates, and they consume a more significant portion of the budget. The sections used in steel structures are I-Beams, channels, angles, double-angles, cylinders, and caps. On the other hand, tubes, tees, elbows, and crosses are the most important components in the piping disciplines.

The companies involved in industrial projects build the BIM models for each discipline (e.g. steel discipline); afterwards, once the models for all the disciplines are built, they are merged into a

single BIM file, which facilitates the collision detection and the quantity take-off process. Since all the disciplines are present in the BIM models, the manual process of selecting a large number of 3D components from the model for previously mentioned component type and generating the projections and exporting them as image files is a time-consuming task.

In this research, an artificially generated dataset instead of using a manually built dataset is proposed. MATLAB is used for generating the artificial dataset. The artificial dataset contains a specific number of images for each component type, which are I-Beams, channels, angles, double-angles, cylinders, caps, tubes, tees, elbows, and crosses. The images in the generated dataset are similar to the actual image obtained from the Autodesk Navisworks, and they include four projections for each 3D model component. Figure 10 shows a sample image for the I-Beam component type of generated dataset for this research.

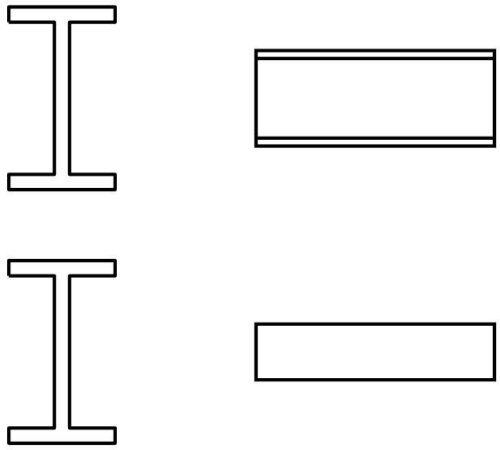


Figure 10. Sample image data of I-Beam from the artificially generated dataset.

Using MATLAB, 100 images for each component type (10 types), and in total, 1000 were generated. The images are in grayscale format with 875 * 656 pixels resolution, and the only variable in generating the dataset was the size of the cross-sections of the components. A sample code for generating dataset for pipes can be found in Appendix A.

3.3. Developing Machine Learning Model

3.3.1. Artificial Neural Networks

There are two fundamental approaches in the field of Artificial Intelligence (AI). The first approach is based on knowledge engineering systems, logic programming and logical reasoning. The second approach mimics the microscopic biological models [60]. Artificial neural networks (ANNs) and genetic algorithms are the prime examples of this latter approach. The field of ANNs was initially configured as an attempt to emulate the way that the brain performs a particular task, by regarding the brain as a highly complex, nonlinear, parallel information processing system [61], [62].

An artificial network is a pool of simple processing units. The communication between the processing units is by sending signals to each other over a large number of weighted connections [63].

Main aspects of a parallel distributed model that can be distinguished are [64]:

- Processing units (‘neurons’ or ‘cells’);
- A state of activation y_k for every unit, which is equivalent to the output of the unit;

- Connections between the units. Generally, each connection is identified by a weight w_{jk} which determines the effect which the signal of the unit j has on unit k;
- A propagation rule, which determines the effective input s_k of a unit from its external inputs;
- An activation function F_k, which determines the new level of activation based on the effective input $s_k(t)$ and the current activation $y_k(t)$ (i.e., the update);
- An external input (bias or offset) θ_k for each unit;
- A method for information gathering (the learning rule);
- An operational environment for the system, providing input and error signals.

Figure 11 illustrates these basics and shows the connections between different units in the network.

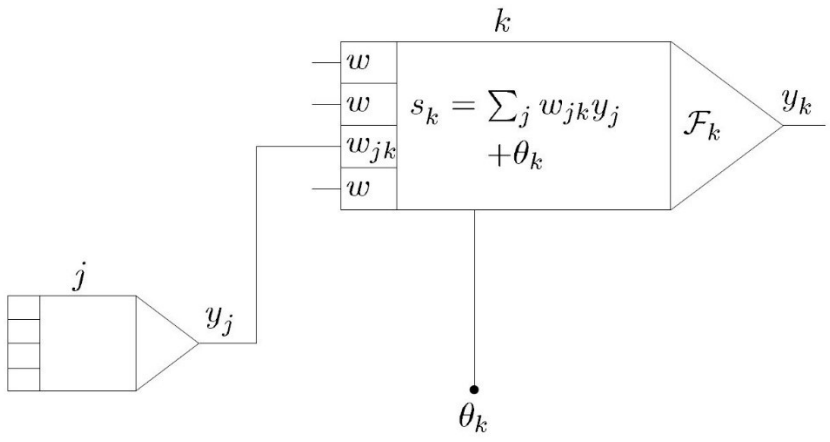


Figure 11. The basic components of an Artificial Neural Network.

In an artificial neural network, each unit in the network performs a relatively simple task: receive input and use it to compute an output signal which is propagated to other units. The inputs can be from neighbours or external sources. Apart from this processing, the second task is the adjustments

of the weights. The system is inherently parallel in the sense that units can carry out their computations simultaneously [63].

Within neural systems, there are three different types of units: *input* units which receive data from outside the neural network, *output* units which send data out of neural network and *hidden* units whose input and output signals remain within the neural network.

The units are usually connected in a way that they provide an additive contribution to the input of the unit with which it is connected. The total input to unit k is simply the weighted sum of the separate outputs from each of the connected units plus a *bias* or *offset* term θ_k [63]:

$$s_k(t) = \sum_j w_{jk}(t)\, y_k(t) + \theta_k(t). \tag{3.1}$$

In order to give the effect of the total input on the activation of the unit, function F_k is needed, which takes the input $s_k(t)$ and the current activation $y_k(t)$, and produces a new value of the activation of the unit k [63]:

$$y_k(t+1) = F_k\big(y_k(t), s_k(t)\big). \tag{3.2}$$

The configuration of a neural network has to be such that desired outputs are produced from the inputs. Various methods are available to set the strength of the connections to existing. One way is using a priori knowledge to set the weights explicitly. Another way is training the neural network by feeding it patterns and letting change its weights according to some learning rules [63].

Learning situations can be categorized into two distinct sorts. There are:

- *Supervised learning* in which the network is trained by providing it with the input that matches output patterns. These input-output pairs can be provided externally, or by a system which contains the network (*self-supervised*) [63].
- *Unsupervised learning* in which the output unit is trained to respond to the patterns that are fed to the input units. In this method, the system is supposed to discover statistically important features of the input population. Unlike the supervised learning method, there is no prior set of categories to classify the patterns into; rather, the system must develop a new representation of the input stimuli [63].

3.3.2. Convolutional Neural Networks

Recently, deep learning has given new power that allows building artificial intelligence (AI) systems that were not possible a few years ago.

The computing infrastructure is based on a hierarchy of perceptions. Each computing layer is characterized in terms of its relation to concepts where the essential layer consists of simple concepts. If a graph is drawn to show how these concepts are built based on each other, the graph will be deep, with many layers. Therefore, this approach is called deep learning, covering several aspects of machine learning [65].

Deep convolutional neural networks (CNNs) are a specific kind of ANNs that uses convolution instead of general matrix multiplication in at least one of their layers [65]. The term “convolution” refers to the mathematical combination of two functions that produce a third function. Unlike the simple neural networks that have one or several hidden layers, CNNs consist

of many layers. Such a feature allows them to compactly represent highly nonlinear and varying functions [66]. CNNs involve many connections, and the architecture is typically comprised of different types of layers, including convolution, pooling and fully-connected layers, and realize a form of regularization [67]. In order to learn complicated features and functions that can represent high-level abstractions (e.g., in vision, language, and other AI-level tasks), CNNs would need deep architectures. Deep architectures, and CNNs, consist of a large number of neurons and multiple levels of latent calculations of non-linearity. Each level of the architecture of CNN represents features at a different level of abstraction, which is defined as a composition of lower-level features [68].

The architecture of the convolutional neural networks is different from regular neural networks. Regular neural networks transform input by putting it through a series of hidden layers within the network. All the layers are made up of a set of neurons, where each layer is fully connected to all neurons in the previous layer. Finally, the last layer, which is a last fully-connected layer — the output layer — represents the predictions [69].

On the other hand, the layers in the convolutional neural networks are organized in 3 dimensions: width, height and depth (RGB). Furthermore, the neurons in one layer do not connect to all the neurons in the next layer; instead, they just connect only to a small region of it. Finally, the final output will be reduced to a single vector of probability scores, organized along the depth dimension [69].

The main benefit of using the traditional CNNs, which is a fully-connected neural network, is the reduced number of parameters that have to be learned [70]. The CNN topology is based on three main concepts, namely: local receptive fields shared weights and spatial or temporal sampling

[54]. It means that CNNs are typically comprised of different types of layers called convolutional layers, whereas each convolutional layer is made of small kernels that allow extracting high-level features in an effective way. The last convolutional layer is fed to fully connected layers. As has been stated before, if CNNs are the reduced number of parameters to be learned, they cause to have much fewer connections and are easier to train [71]. Consequently, this particular kind of neural network assumes that it should learn filters in a data-driven fashion as a mean to extract features that describe the inputs [67].

A standard CNN model has a structure consisting of the input layer, alternating convolutional layers, pooling or subsampling layers and non-linear layers [69]. Accordingly, with a complex layer terminology, one convolutional net or convolutional layer is composed of a convolutional stage, detector stage, and pooling stage [72]. This means that each convolutional layer has more than one stage. Therefore, each stage of the convolutional layer can be set apart, and every step of processing it can be ruled in its rights. Convolutional layers are typically interspersed with sub-sampling layers to reduce computational time and gradually build up further spatial and configural invariance [68].

In the next section, different layers of a CNN and the architecture used for CNN in this research will be elaborated.

3.3.3. Proposed CNN Model

In the developed model, a convolutional neural network model with a simple architecture of 2 convolutional layers is chosen, as the types of images that have to be classified are simple.

Usually, the input is a multidimensional array of data where data are fed to the network [67]. Input data can be, i.e., patterns, image pixels or their transformation, time series or video signals. In the present model, the input data are fed to the network in $64 \times 64 \times 1$ arrays, which are the values for each pixel in the images as they are in grayscale. In order to decrease the amount of the calculations and complexity in the model, all the images were generated in grayscale; therefore, there is one value for each pixel instead of having three values in RGB format. Different image resolutions were tested in the training stage of the model, and 64×64 pixel resolution had the best performance in terms of the calculation time and the overall validation accuracy. Figure 12 shows the structure for the proposed CNN model.

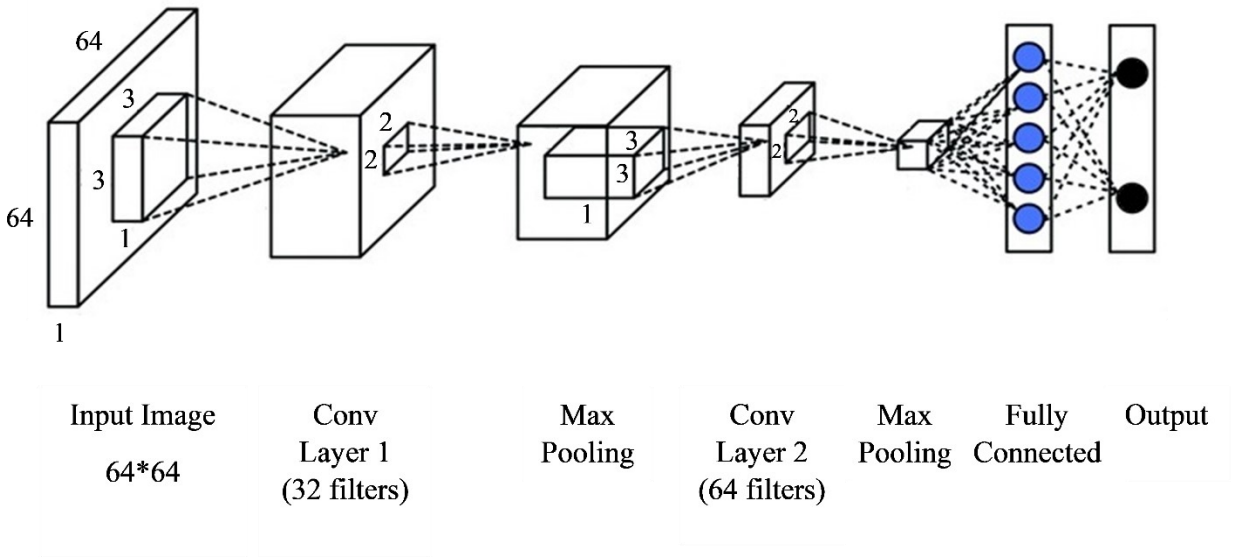


Figure 12. Structure of the proposed CNN.

Convolutional layers are the main building block of CNN. The prime purpose of convolution is to extract distinct features from the input [69]. These layers are comprised of a series of filters or learnable kernels which aim to extract local features from the input, and each kernel is used to calculate a feature map or kernel map [73].

The first convolutional layer extracts low-level meaningful features such as edges, corners, textures and lines. The next convolutional layer extracts higher-level features, but the highest-level features are extracted in the last convolution layer [74]. Kernel size refers to the size of the filter, that convolves around the feature map while the amount by which the filter slides (sliding process) is the stride. It controls how the filter convolves around the feature map of the input. As a result, the filter convolves around the different layers of the input feature map by sliding one unit each time [65].

The number of filters in each convolutional layer depends on (1) the complexity of the dataset, and (2) the depth of the neural network. Since the dataset used for the training and the architecture of the proposed neural network are not complex, in the first and second convolutional layer, the model learns a total of 32 and 64 filters, respectively. These values are recommended practically for simple models. The number of filters in each layer can be increased as the model gets more complex.

The next required parameters that need to be provided to the model are the kernel size and the stride value. The kernel size is a matrix specifying the width and height of the convolutional window, and the stride parameter is a 2-tuple of integers, specifying the “step” of the convolution along the x and y-axis of the input image. The typical values for the kernel size include: (1, 1), (3, 3), (5, 5), and (7, 7). Kernel sizes of (5, 5), and (7, 7) are used for images with pixel resolutions

higher than 128×128. For the image resolution of 64×64 used in the model, the best option for the kernel size is (3, 3), as it helps learn larger spatial filters and reduces the volume size more quickly compared to the kernel size of (1, 1). The stride value used in the model is the default value of (1, 1), which implies that the filter takes a 1-pixel step to the right, and again the filter is applied to the input image. This process is performed until the far-right border of the image is reached, in which the filter is moved one pixel down and then starts again from the far left.

Padding is another essential feature of CNNs that gives the option to make input data wider so that it does not miss any features that are in the corners [69]. In order to reduce the calculation time, zero-padding was not used in the model, and the spatial dimensions were reduced via the natural application of convolution.

Non-linear layers are used to detect each linear activation through a nonlinear activation function. In other words, linear activation introduces the non-linearity into neural networks and allows learning more complex models [69].

There are several nonlinear activation functions. The standard way to model the output f of a neuron as a function of its input x is with $f(x) = sigmoid(x)$, $tanh(x)$, or Rectified Linear Unit (ReLU) [72]. The last one is preferable since it makes the training process several times faster than its equivalents. Some authors adopt the sigmoid function at all activation stages due to its simplicity [69]. ReLU applies the function $y = \max(x, 0)$. It increases the nonlinearity in the properties of the decision function and the overall network without affecting the receptive fields in the convolutional layer. Using ReLU, it is possible to speed up the training of CNNs by keeping up the gradient more or less constant at all network layers [71].

The pooling layer reduces the resolution of the previous feature maps through compressing features and the computational complexity of the network [75]. It adjusts the features robust to disorder and noise. Another reason to use the pooling layer is to make it robust to small variations for previously learned features. As a result, pooling ensures that the network focuses on the most important patterns. In general, a pooling layer downsamples the input map and reduces the dimensionality of the feature maps used by the following layers [67], [68].

In the pooling stage, the inputs are split into regions with the size of $R \times R$ to produce one output from each region. If a given input with a size of $W \times W$ is fed to the pooling layer, the output size P is obtained by [76]:

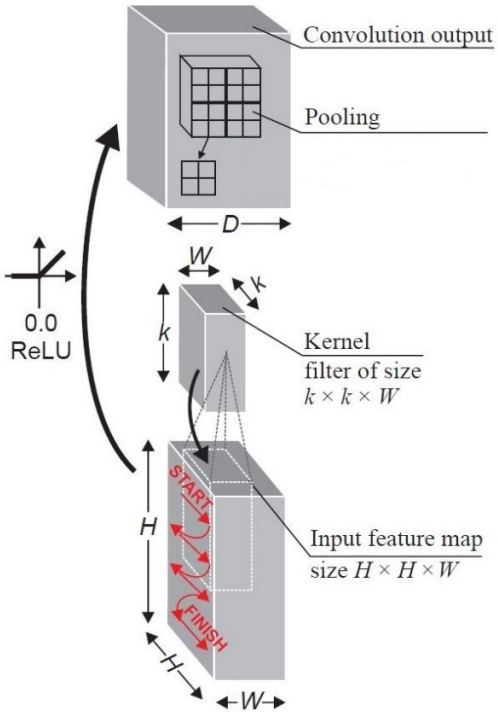
$$P = \left\lfloor \frac{W}{R} \right\rfloor \tag{3.3}$$

Pooling can be max pooling, an average of a rectangular neighbourhood, and pooling by downsampling.

The max-pooling action addresses the maximum output within a rectangular neighbourhood. Max pooling outputs only the maximum number in each kernel, thus reducing the feature map size. Max pooling introduces invariance. For maximum pooling, the maximum value of the four values is selected. For average pooling, the average of the four values is selected. Max pooling extracts the most important features like edges, whereas average pooling extracts features smoothly, as it takes all the values in the pool into consideration. For image data, which are the inputs for our model, max-pooling extracts the extreme features, which results in more accurate models in image classification tasks.

In general, the feature extraction using CNNs consists of multiple similar steps, and each step is made of three cascading layers: convolution layer, activation layer and pooling function.

Figure 13 exhibits the process of 3D convolution used in CNNs.



H – input height and width

W – input depth, kernel width

K – kernel height and width

s – kernel stride

n – number of kernels

D - #output feature maps

Figure 13. The convolution process.

The input with the size of $H \times H \times W$ is convoluted with p number of kernels, where the kernel size is $k \times k \times W$. Each kernel convolving with an input feature map produces one output feature map, and p kernel produces p feature maps, independently. Each kernel is moved to start from the top-left corner of the input feature map to the top-right corner element at a time. Then the kernel

shifts one element downward, takes a left-side position and moves towards the right-side position. This process is finished when the kernel reaches the bottom-right position.

For instance, for the case when the input is $H = 64$ and $k = 3$, there are 62 unique positions from left to right, and 62 unique positions from the top to the bottom that the kernel can take. Each feature in the convolution output will contain 62×62, i.e., $(H - k + 1) \times (H - k + 1) = (64 - 3 + 1) \times (64 - 3 + 1)$ elements. To create one element of one output feature, $k \times k \times W$ operations are required.

From the considerations mentioned above, it can be concluded that a new feature map is typically generated by sliding a filter over the input and computing the dot product (which is similar to the convolution operation), followed by a non-linear activation function to introduce non-linearity into the model.

The fully connected layer is the last stage of the topology of CNNs, consisting of a generic multi-layer network. The last layer is a fully connected 1D layer to all activations in the previous layer [68]. From these layers, the features can be extracted to train another classifier. To specify how the network training penalizes the deviation between the predicted and the true labels, different loss functions can be used, e.g., softmax, and sigmoid functions [67]. Softmax function can be used for multi-class classification tasks in logistic regression models, while the sigmoid function is used for binary classification models. Since there are ten different classes in our model, the softmax function will be used for the fully connected layer. The output of the last layer with softmax function is a vector of values for the probability of each label with the target label having the highest probability.

The development and training of the model are done using Keras, which is a high-level neural networks API, written in Python that is capable of running on top of TensorFlow. The accuracy of the algorithm is dependent on the resolution of the input images extracted from Autodesk Navisworks, the size of the dataset used for retraining the model, and the orientation of the components. For training the CNN model, four different scenarios were tested, in which the image resolution was increased. The resolutions of 16×16, 32×32, 64×64, and 128×128 pixels were tested. As shown in Figure 14, as the image resolution increases, at the same time, training accuracy and computation time increase. After testing these scenarios, the CNN model using 64×64 pixel images was opted to be used, which has 100% accuracy and takes 42 seconds to train the model. The code for the proposed CNN model can be found in Appendix B.

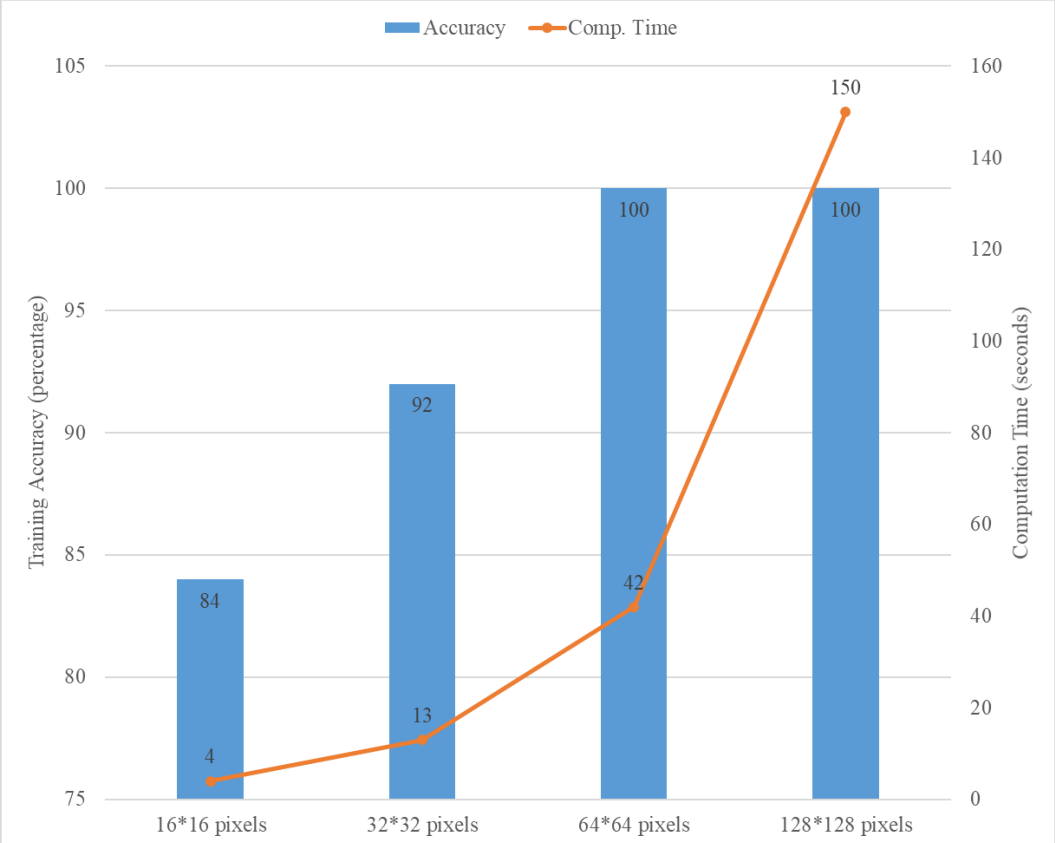


Figure 14. Training accuracy for four scenarios.

In order to be consistent, the trained model was saved and transferred to the Microsoft Visual Studio environment, as the plugin for Autodesk Navisworks is developed using the same environment. In addition, to reduce the training time each time the model is used for classification, transfer learning, which is the process of saving and reusing the model to predict new instances was used.

Transfer learning is the process of using the knowledge gained while solving one problem and applying it to a different but related problem. Because the developed model using Keras has already been trained on a large number of images, internally, it contains all the image features that are needed for image identification. These internal image features in the model can be used to train a new model with far fewer instances, which reduces the calculation time.

4. Implementation of the Proposed Method

4.1. The 3D Viewing Model Platform Navisworks

In the present research, all the design models are converted and then imported into Autodesk Navisworks. Next, Navisworks files are delivered to the general contractors. AutoCAD Navisworks is one of the most widely used 3D viewing platforms in the construction industry [37]. In Navisworks, the users can integrate models from different sources; the source files can be 3D BIM model from different design platforms, such as Tekla, SmartPlant, and Autodesk Revit. The users can view, comment, highlight, hide, and measure the Navisworks models using tool packages. The tool packages also come with plug-ins to implement clash detections, 4D simulation and visualization rendering. Navisworks models are intended only for viewing; the model components cannot be edited, duplicated, or deleted. This limitation reduces the risk of source model providers in the ownership of their intellectual property.

4.2. API of Autodesk Navisworks

The application programming interface (API) is a collection of routines, tools, and protocols used to develop the application software. For the convenience of the users, some information and functions are hidden from the interactive user interface. API can help to access the hidden information and functions. The API provides tools that can be used to develop customized programs for different purposes, which add to applications and functions of the software.

Navisworks provides API such as .NET API, NwCreate API, and COM API. The COM API is used to manipulate models and documents. It has a plug-in and automation and provides ActiveX controls. The Navisworks API supports C# and Visual Basic language. Therefore using the Autodesk Navisworks API and C# language in the Visual Studio environment, the custom functions that are not built-in can be developed. In this research, the method was developed using COM API and .NET based on C# language.

4.3. Access Model Components in AutoCAD Navisworks

In a Navisworks file, the model of the project is a combination of sub-models imported from different sources. Each sub-model can be subdivided into the lower level model parts. The relationships of the model components are listed as a tree structure, which can be found in a panel in Navisworks as the "Selection Tree." The lowest level of components in the selection tree will be selected as the basic components such as tubes, I-beams, etc. for the quantity take-off. In the present research, to improve the accuracy and reduce the complexity of the developed algorithm, the algorithm will be applied to each basic component. The information about the type of each basic component will be predicted using the CNN model.

The components for the quantity take-off process need to be found from the selection tree. In industrial construction models, millions of components exist in the selection tree. The top node in the selection tree is defined as the root. A lower-level node directly connected to the top node is defined as the child of the node, and the converse notion of the child node is defined as a parent node. Navisworks will start from the root in the selection tree. After every node is visited at the

current level, the program will continue to visit the next lower level. Furthermore, by the end, all the components in the selection tree will be visited.

4.4. Developed Autodesk Navisworks Plugin

In this research, Autodesk Navisworks Plugin and an image classifier were developed as the industrial contribution. There are two steps in identifying the labels of the components in BIM models, which are generating and saving the snapshots and classifying the components.

The first step can be done using the developed plugin, after opening the BIM model in the Autodesk Navisworks and selecting the components that need to be labelled. Firstly, the plugin isolates each of the components within the selection and generates a viewpoint of them in 4 projections, namely: top, side, front, and back view. Next, it exports each viewpoint as an image file with a *".png"* file format and merges all the four projections in a single image with a tabular format. Figure 15 shows the environment of the developed plugin. Appendix C and D includes the codes for developing the Naviswork plugin.

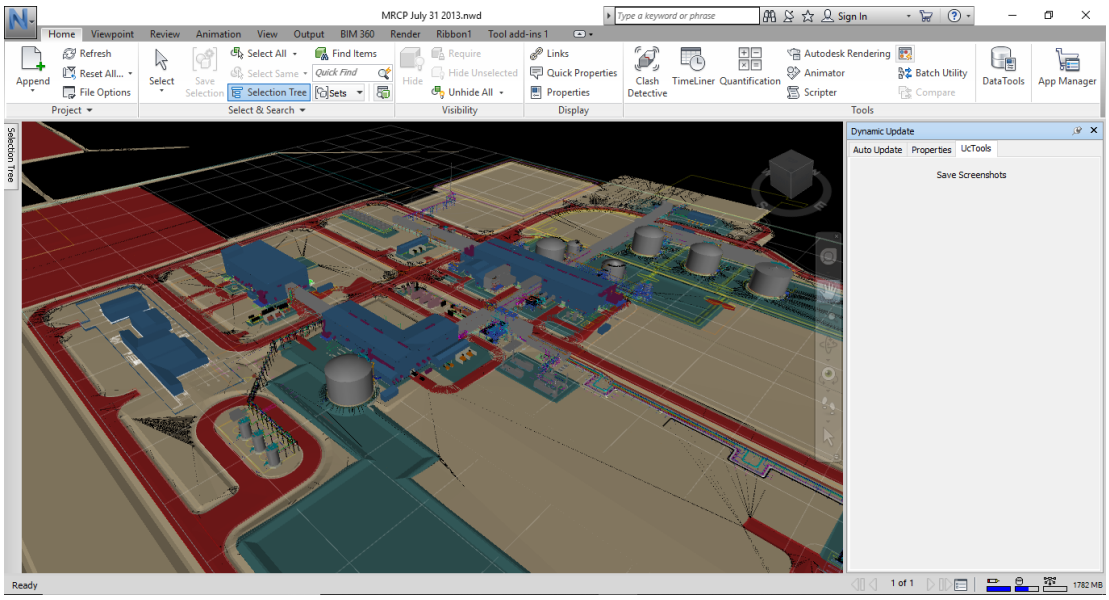


Figure 15. The environment of the developed Autodesk Navisworks Plugin.

The next step is classifying the unlabeled components, which is done by using the developed CNN classifier. The inputs of the classifier are the saved images using the plugin, and the output is an excel file containing the ID of each component in the BIM model and the label provided by the classifier. The process is summarized as the workflow shown in Figure 16.

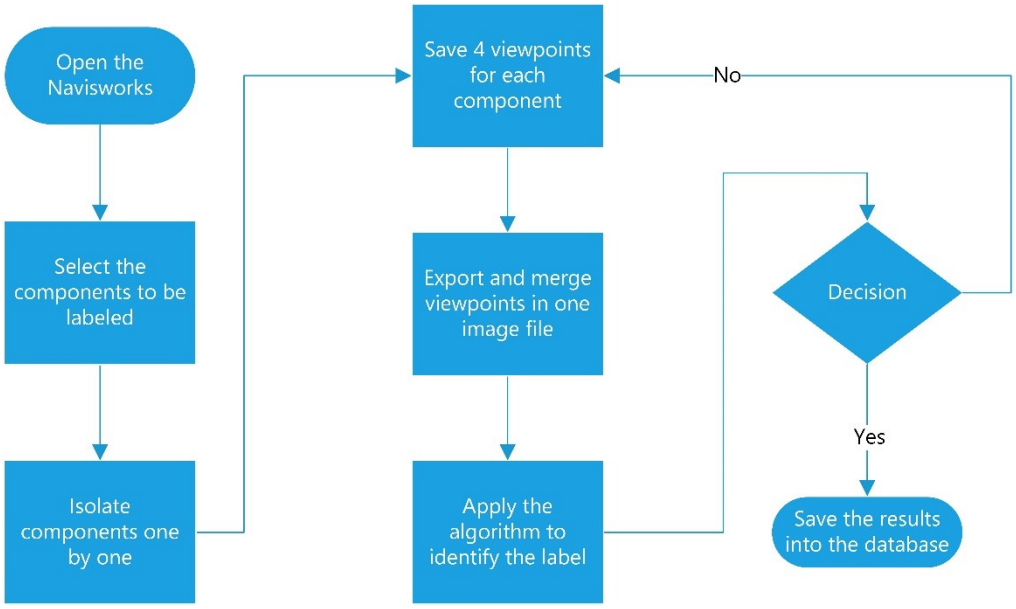


Figure 16. The workflow of the proposed method.

5. Method Validation

This section contains a summary of our analysis of the method performance using three real-world BIM models obtained from a major contractor in Canada. These models have been created by three different engineering firms around the world. Consequently, each model has different labels and colour-coding. The contractor performs a preliminary manual estimate for these projects during the design stage.

The number of items in each model varies from a half-million to three-quarters of a million items; this includes items from different disciplines such as mechanical, electrical, and structural. A visual inspection showed that steel items included angles for trusses and bracing, and H sections for columns and beams. In addition, the size and orientation changed significantly from one item to the next. Figure 17 shows a full sample model for one of the projects.

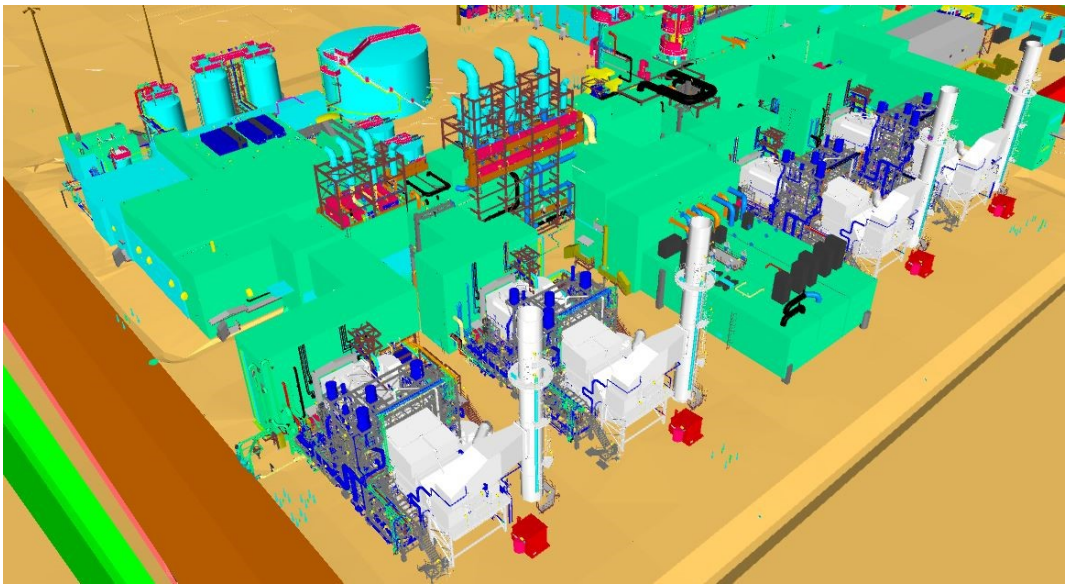


Figure 17. Sample BIM model.

A random sample of components was drawn and identified them manually and then ran the algorithm independently using the same components and compared the results. The sample size for testing the model was calculated based on the following equation with a 99% confidence level and a 5% margin of error. Since the population of the steel and piping disciplines were not known, the value of p was decided to be 0.5 to have a conservative sample size. Using the equation (5.1) the sample size of testing was 666, which was rounded up to 1000 to be conservative as the population is more than millions of components. From these 1000 sample, 334 of them are the components that are unidentified to the classifier, meaning that the classifier is unable to classify them as there was dataset of them fed to the classifier during training phase. The sample was drawn from the three models using a simple random method in which all items had an equal chance to be drawn. The sample contained steel and piping components with different shapes.

$$n = \frac{Z^2 \times p \times q}{E^2} \tag{5.1}$$

After drawing the sample, each item was manually identified and then identified the sample by the algorithm several times using different image resolutions. For each scenario, the success rate and the average computation time were recorded.

In order to calculate the average computation time, the total time required to identify the total sample was calculated, then divided that by the number of sections in the sample; this is more accurate than calculating the time required to identify one section because of the fixed time required to load the reference database into the memory.

5.1. Results

In order to validate the model after training, the model was tested using 1000 randomly selected components from three real-world BIM models. Table 1 shows the confusion matrix for the validation. A confusion matrix is a table that is usually used to describe the performance of a classification model on a set of test data for which the actual values are known. In the confusion matrix table, the columns are predicted values, and the rows are actual values. There are three main metrics for the performance of classification models that can be calculated based on the confusion matrix that are accuracy, precision, and recall. While accuracy represents the overall performance of the classification model, precision and recall are calculated for each class separately. Accuracy can be calculated as:

$$Accuracy = \frac{TP+TN}{TP+TN+FP+FN} \tag{5.1}$$

To calculate the precision, the total number of correctly classified positive examples are divided by the total number of predicted positive examples. High Precision indicates that an example labelled as positive is indeed positive.

$$Precision = \frac{TP}{TP+FP} \tag{5.2}$$

Recall can be defined as the ratio of the total number of correctly classified positive examples divided to the total number of positive examples. High Recall indicates the class is correctly recognized. Since higher recall means that most of the positive examples are correctly recognized, it is more important than the precision metric in this research.

$$Recall = \frac{TP}{TP+FN} \tag{5.3}$$

In the above-mentioned formulas, *TP, TN, FP, FN* indicate True Positive, True Negative, False Positive and False Negative, respectively.

In this research, the accuracy of the classification model is 91%, and class precision and recall for different component types can be found in Figure 18 and Figure 19. In the calculation of the accuracy, recall and the precision the unidentified components were included, as the classifier could successfully label them as unidentified. Also, there were some components from other classes that were identified as unidentified, which is because of low probability. There are some classes indicating 100% recall, which means that the classifier will label them correctly all time, which results in less supervision. In addition, there are four classes (I-Beam, Angle, Double angle, and Channel) that have less recall than the other classes. The reseaon behind the low recall is that since the classifier is sensitive to rotations, and these classes are similar, the classifier was unable to detect the arbitrary rotations.

	I-Beam	Angle	Double Angle	Channel	Cylinder	Cap	Tube	Elbow	Tee	Cross	Unidentified
I-Beam	156	3	5	3	0	0	0	0	0	0	8
Angle	7	27	9	1	0	0	0	0	0	0	2
Double Angle	4	5	24	2	0	0	0	0	0	0	5
Channel	3	2	1	29	0	0	0	0	0	0	0
Cylinder	0	0	0	0	12	0	0	0	0	0	0
Cap	0	0	0	0	0	9	0	0	0	0	0
Tube	0	0	0	0	0	0	232	0	0	0	0
Elbow	0	0	0	0	0	0	0	59	0	0	0
Tee	0	0	0	0	0	0	0	0	44	0	0
Cross	0	0	0	0	0	0	0	0	0	14	0
Unidentified	0	0	0	0	0	0	0	0	0	0	334

Table 1. Confusion matrix of the model validation.

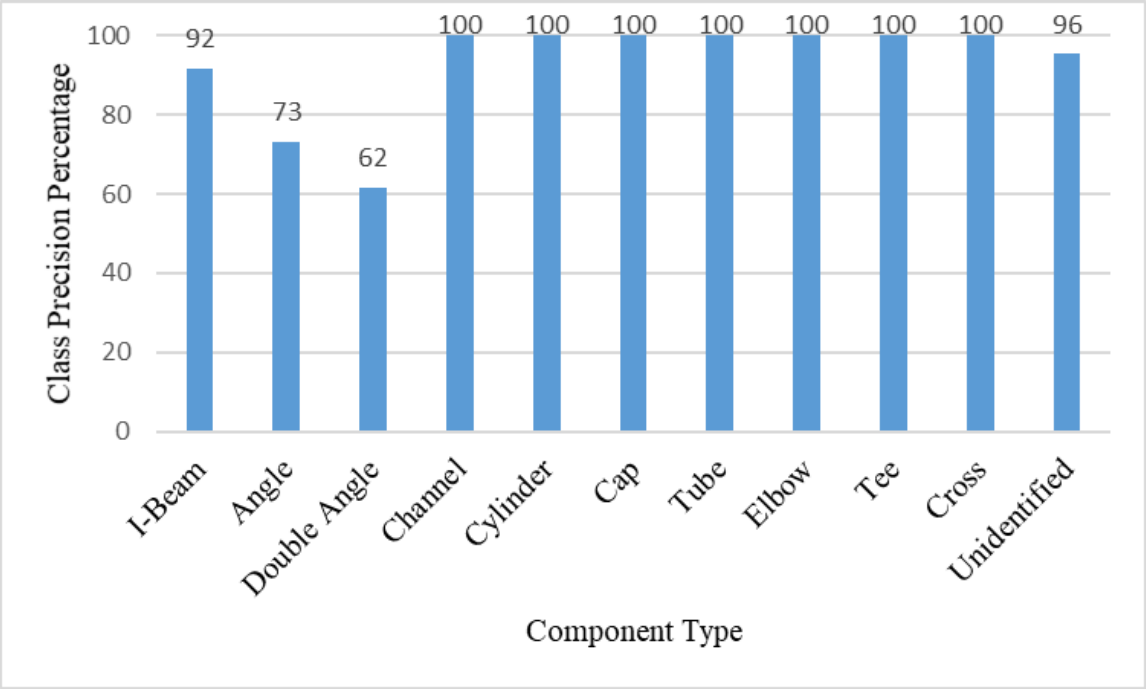


Figure 18. Class precision of the classification model.

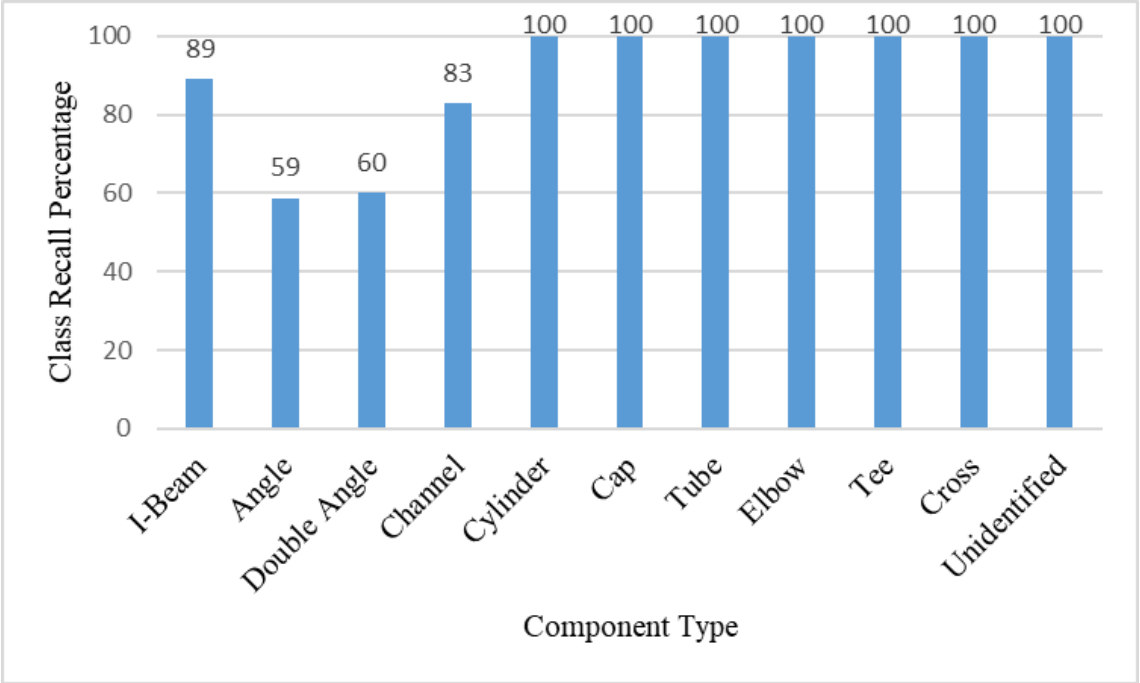


Figure 19. Class recall for the classification model.

6. Discussion and Conclusion

6.1. Conclusion

Building Information Modeling (BIM) has changed our way of dealing with construction projects. It serves as a data store that can capture attributes other than geometrical objects. This allows engineers and contractors to work mainly with one source of the data that is expected to provide all information related to a project.

However, the current data flow practice between engineering firms and contractors and the usage of customized BIM solutions in industrial projects limits the potentials of BIM, as there is no consensus on the naming convention, and the meta-data is not fully described in BIM models, especially during the early stages of projects. This leads to what is known as a "Dump Model," which can be inspected visually but is hard or impossible to utilize for repetitive tasks, such as quantity take-off, that are needed for planning.

This research aims to leverage information usage in BIM models by automatically completing and validating missing data in the BIM models. Our work used image classification techniques to find the correct type of steel and piping component in the model.

After reviewing 3D image classification algorithms in the literature, the view-based convolutional neural networks was selected. View-based methods project 3D objects into multiple 2D views; then, the classification is conducted using the features from 2D CNNs. In the first step, a Navisworks plugin was developed to export the viewpoint as images and merge all four views into one image with a tabular format. Then, an artificial dataset of images was generated for training

the CNN model. After testing the performance of the trained model, it was transferred to Visual Studio to classify the images exported from the Navisworks.

It was tested using data from three real BIM models for oil and gas projects that have been executed in Alberta, Canada, during the last decade. The average budget for each project is C$750 million. Each model is a typical "Dump Model" that contains 3D components but without enough attributes to provide an accurate description. These early-stage models, which are commonly used in fast-tracked projects, cannot be easily categorized by component type, let alone categorized by their classes. The results show that the proposed classification method is able to achieve 91% accuracy.

6.2. Research Contribution

The work contributions have been categorized into academic and industrial.

6.2.1. Academic Contribution

The main academic contribution of the study showed that the convolutional neural networks could be used to help the engineering firms with recognizing the type of steel and piping components in unlabeled BIM models based on their projections. The results showed that the model could identify the labels with 91% accuracy.

6.2.2. Industrial Contribution

The industrial contribution of the study introduced is a tool that can perform classification methods on BIM objects during the early stages of a project to help with the preliminary quantity take-off process.

6.3. Limitations and Future Work

Although this research manages to provide an alternative to the manual process with acceptable accuracy during the early stages of the projects, there are limitations that can be summarized as follows:

1. The proposed method will fail to detect arbitrary shapes as it only compares unidentified components to a reference set of components.
2. The algorithm will fail to detect the label for the components with arbitrary orientations because of the limited dataset.
3. The method is intended only for preliminary estimates and cannot be used, for example, to formulate the bill of materials.

This research can be extended in two different directions. One direction would be to apply the same methodology to other component types, i.e. mechanical. Applying the methodology to mechanical components would provide an interesting opportunity to compare how the algorithm performs with different disciplines.

Another direction for this research is using more sophisticated architecture for the convolutional neural network and measuring the difference in accuracy and processing time.

Bibliography

[1] H. Tanaka, “Toward Project and Program Management Paradigm in the Space of Complexity: A Case Study of Mega and Complex Oil and Gas Development and Infrastructure Projects,” *Procedia - Social and Behavioral Sciences*, vol. 119, pp. 65–74, 2014, doi: 10.1016/j.sbspro.2014.03.010.

[2] C. L. Macken, S.-H. Lee, N. Gcr, and S. R. Thomas, “Impacts of Design / Information Technology Building and Industrial Projects,” *Journal Article*, no. Impacts of Design / Information Technology, pp. 1–51, 2001, [Online]. Available: https://ws680.nist.gov/publication/get_pdf.cfm?pub_id=860090.

[3] C. Eastman, P. Teicholz, R. Sacks, and K. Liston, *BIM Handbook: A Guide to Building Information Modeling For Owners*. 2011.

[4] S. Azhar, “Building information modeling (BIM): Trends, benefits, risks, and challenges for the AEC industry,” *Leadership and Management in Engineering*, vol. 11, no. 3, pp. 241–252, 2011, doi: 10.1061/(ASCE)LM.1943-5630.0000127.

[5] D. Bryde, M. Broquetas, and J. M. Volm, “The project benefits of Building Information Modelling (BIM),” *International Journal of Project Management*, vol. 31, no. 7, pp. 971–980, Oct. 2013, doi: 10.1016/J.IJPROMAN.2012.12.001.

[6] A. Wong, F. K. W. Wong, and A. Nadeem, “Government roles in implementing building information modelling systems Comparison between Hong Kong and the United States,” no. March 2016, 2011, doi: 10.1108/14714171111104637.

[7] A. Porwal and K. N. Hewage, “Automation in Construction Building Information Modeling (BIM) partnering framework for public construction projects,” *Automation in Construction*, vol. 31, pp. 204–214, 2013, doi: 10.1016/j.autcon.2012.12.004.

[8] Y. Jung and M. Joo, “Automation in Construction Building information modelling (BIM) framework for practical implementation,” *Automation in Construction*, vol. 20, no. 2, pp. 126–133, 2011, doi: 10.1016/j.autcon.2010.09.010.

[9] G. Aranda-mena and T. M. Froese, “Building information modelling demystified : Does it make business sense to adopt BIM ?,” no. June, 2009, doi: 10.1108/17538370910971063.

[10] N. Han, Z. Yue, and Y. Lu, “Collision Detection of Building Facility Pipes and Ducts Based on BIM Technology,” *Advanced Materials Research*, vol. 346, pp. 312–317, Sep. 2011, doi: 10.4028/www.scientific.net/AMR.346.312.

[11] A. Ahmad Latiffi, S. Mohd, N. Kasim, and M. S. Fathi, “Building Information Modeling (BIM) Application in Malaysian Construction Industry,” *International Journal of Construction Engineering and Management*, vol. 2, p. 6, Jan. 2013, doi: 10.5923/s.ijcem.201309.01.

[12] R. Amor and H. Ma, “Preservation of Meaning in Mapped IFCs,” Sep. 2006.

[13] T. Rujirayanyong and J. J. Shi, “A project-oriented data warehouse for construction,” *Automation in Construction*, vol. 15, no. 6, pp. 800–807, 2006, doi: https://doi.org/10.1016/j.autcon.2005.11.001.

[14] K. W. Chau, M. Anson, and J. P. Zhang, “4D dynamic construction management and visualization software: 1. Development,” *Automation in Construction*, vol. 14, no. 4, pp. 512–524, 2005, doi: https://doi.org/10.1016/j.autcon.2004.11.002.

[15] T. W. Kang and C. H. Hong, “A study on software architecture for effective BIM/GIS-based facility management data integration,” *Automation in Construction*, vol. 54, pp. 25–38, 2015, doi: https://doi.org/10.1016/j.autcon.2015.03.019.

[16] Y. Chen and J. M. Kamara, “A framework for using mobile computing for information management on construction sites,” *Automation in Construction*, vol. 20, no. 7, pp. 776–788, 2011, doi: https://doi.org/10.1016/j.autcon.2011.01.002.

[17] M. Yavartanoo, E. Y. Kim, and K. M. Lee, “SPNet: Deep 3D Object Classification and Retrieval Using Stereographic Projection BT - Computer Vision – ACCV 2018,” 2019, pp. 691–706.

[18] A. M. Hammad, “An Integrated Framework for Managing Labour Resources Data In Industrial Construction Projects : a Knowledge Discovery In Data (kdd) Approach,” 2009.

[19] J. Song, C. T. Haas, C. Caldas, E. Ergen, and B. Akinci, “Automating the task of tracking the delivery and receipt of fabricated pipe spools in industrial projects,” *Automation in Construction*, vol. 15, no. 2, pp. 166–177, 2006, doi: https://doi.org/10.1016/j.autcon.2005.03.001.

[20] H. Timo, G. Ju, and F. Martin, “Areas of Application for 3D and 4D Models on Construction Projects,” *Journal of Construction Engineering and Management*, vol. 134, no. 10, pp. 776–785, Oct. 2008, doi: 10.1061/(ASCE)0733-9364(2008)134:10(776).

[21] S. STAUB–FRENCH, M. FISCHER, J. KUNZ, K. O. S. ISHII, and B. PAULSON, “A feature ontology to support construction cost estimating,” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, vol. 17, no. 2, pp. 133–154, 2003, doi: DOI: 10.1017/S0890060403172034.

[22] M. Ali, Y. Mohamed, H. Taghaddos, and R. Hermann, “BIM obstacles in industrial projects : a contractor perspective.” Proceedings of The Canadian Society for Civil Engineering 5th International/11th Construction Specialty Conference, 2015, [Online]. Available: https://open.library.ubc.ca/collections/52660/items/1.0076376.

[23] C. Anumba, J. Pan, R. Issa, and I. Mutis, “Collaborative Project Information Management in a Semantic Web Environment,” *Engineering, Construction and Architectural Management*, vol. 15, pp. 78–94, Jan. 2008, doi: 10.1108/09699980810842089.

[24] N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani, “Three-dimensional shape searching: state-of-the-art review and future trends,” *Computer-Aided Design*, vol. 37, no. 5, pp. 509–530, 2005, doi: https://doi.org/10.1016/j.cad.2004.07.002.

[25] H. Tommy, C. Hamish, T.-H. Linh, and L. D. F., “Point Cloud Data Conversion into Solid Models via Point-Based Voxelization,” *Journal of Surveying Engineering*, vol. 139, no. 2, pp. 72–83, May 2013, doi: 10.1061/(ASCE)SU.1943-5428.0000097.

[26] A. Mostafa, M. Yasser, and H. Rick, “Automated Recognition of Unlabeled Items in BIM Models,” *Construction Research Congress 2016*. pp. 2207–2217, Apr. 12, 2020, doi: doi:10.1061/9780784479827.220.

[27] P. Tang, D. Huber, B. Akinci, R. Lipman, and A. Lytle, “Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques,” *Automation in Construction*, vol. 19, no. 7, pp. 829–843, 2010, doi: https://doi.org/10.1016/j.autcon.2010.06.007.

[28] H. Woo, E. Kang, S. Wang, and K. H. Lee, “A new segmentation method for point cloud data,” *International Journal of Machine Tools and Manufacture*, vol. 42, no. 2, pp. 167–178, 2002, doi: https://doi.org/10.1016/S0890-6955(01)00120-1.

[29] J. C. Carr *et al.*, “Reconstruction and Representation of 3D Objects with Radial Basis Functions,” in *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, 2001, pp. 67–76, doi: 10.1145/383259.383266.

[30] F. Remondino, “From point cloud to surface: The modeling and visualization problem,” *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 34, Mar. 2004, doi: 10.3929/ethz-a-004655782.

[31] A. Gruen and D. Akca, “Least squares 3D surface and curve matching,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 59, no. 3, pp. 151–174, 2005, doi: https://doi.org/10.1016/j.isprsjprs.2005.02.006.

[32] M. Pauly, R. Keiser, L. Kobbelt, and M. Gross, “Shape Modeling with Point-Sampled Geometry,” *ACM Transactions on Graphics*, vol. 2, pp. 52–66, Jul. 2003, doi: 10.1145/882262.882319.

[33] P. Axelsson, “Processing of laser scanner data—algorithms and applications,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 54, no. 2, pp. 138–147, 1999, doi: https://doi.org/10.1016/S0924-2716(99)00008-8.

[34] I. Brilakis *et al.*, “Toward automated generation of parametric BIMs based on hybrid video and laser scanning data,” *Advanced Engineering Informatics*, vol. 24, no. 4, pp. 456–465, 2010, doi: https://doi.org/10.1016/j.aei.2010.06.006.

[35] H.-G. Maas and G. Vosselman, “Two algorithms for extracting building models from raw laser altimetry data,” *ISPRS Journal of Photogrammetry and Remote Sensing*, vol. 54, no. 2, pp. 153–163, 1999, doi: https://doi.org/10.1016/S0924-2716(99)00004-0.

[36] X. Xiong, A. Adan, B. Akinci, and D. Huber, “Automatic creation of semantically rich 3D building models from laser scanner data,” *Automation in Construction*, vol. 31, pp. 325–337, 2013, doi: https://doi.org/10.1016/j.autcon.2012.10.006.

[37] yi feng Chang and S.-G. Shih, “BIM-based Computer-Aided Architectural Design,” *Computer-Aided Design and Applications*, vol. 10, pp. 97–109, Aug. 2013, doi: 10.3722/cadaps.2013.97-109.

[38] P. Brown, “CAD : Do Computers Aid the Design Process After All ?,” *The Stanford Journal of Science, Technology and Society*, vol. 2, no. 1, 2009.

[39] X. Ye, W. Peng, Z. Chen, and Y.-Y. Cai, “Today’s students, tomorrow’s engineers: an industrial perspective on CAD education,” *Computer-Aided Design*, vol. 36, no. 14, pp. 1451–1460, 2004, doi: https://doi.org/10.1016/j.cad.2003.11.006.

[40] J. Pan and C. J. Anumba, “Semantic-Discovery of Construction Project Files,” *Tsinghua Science & Technology*, vol. 13, pp. 305–310, 2008, doi: https://doi.org/10.1016/S1007-0214(08)70166-4.

[41] T. Funkhouser *et al.*, “A Search Engine for 3D Models,” *ACM Transactions on Graphics*, vol. 22, no. 1, pp. 83–105, 2003.

[42] A. Cardone, S. Gupta, and M. Karnik, “A Survey of Shape Similarity Assessment Algorithms for Product Design and Manufacturing Applications,” *J. Comput. Inf. Sci. Eng.*, vol. 3, pp. 109–118, Jun. 2003, doi: 10.1115/1.1577356.

[43] J. W. H. Tangelder and R. C. Veltkamp, “A survey of content based 3D shape retrieval methods,” *Multimedia Tools and Applications*, vol. 39, no. 3, p. 441, 2007, doi: 10.1007/s11042-007-0181-0.

[44] C. Ruizhongtai Qi, H. Su, M. NieBner, A. Dai, M. Yan, and L. Guibas, “Volumetric and Multi-view CNNs for Object Classification on 3D Data,” in *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2016, pp. 5648–5656, doi: 10.1109/CVPR.2016.609.

[45] B. K. P. Horn, “Extended Gaussian images,” *Proceedings of the IEEE*, vol. 72, no. 12, pp. 1671–1686, 1984, doi: 10.1109/PROC.1984.13073.

[46] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin, “Shape distributions,” *ACM Transactions on Graphics*, vol. 21, no. 4, pp. 807–832, 2002, doi: 10.1145/571647.571648.

[47] S. Chaudhuri and V. Koltun, “Data-Driven Suggestions for Creativity Support in 3D Modeling,” *ACM Trans. Graph.*, vol. 29, no. 6, Dec. 2010, doi: 10.1145/1882261.1866205.

[48] A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov, “Shape Google: Geometric Words and Expressions for Invariant Shape Retrieval,” *ACM Trans. Graph.*, vol. 30, no. 1, Feb. 2011, doi: 10.1145/1899404.1899405.

[49] I. Kokkinos, M. M. Bronstein, R. Litman, and A. M. Bronstein, “Intrinsic shape context descriptors for deformable shapes,” in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2012, pp. 159–166, doi: 10.1109/CVPR.2012.6247671.

[50] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. van Gool, “Hough Transform and 3D SURF for Robust Three Dimensional Classification,” in *Proceedings of the 11th European Conference on Computer Vision: Part VI*, 2010, pp. 589–602.

[51] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz, “Rotation Invariant Spherical Harmonic Representation of 3D Shape Descriptors,” in *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, 2003, pp. 156–164.

[52] D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung, “On Visual Similarity Based 3D Model Retrieval,” *Computer Graphics Forum*, vol. 22, no. 3, pp. 223–232, 2003, doi: 10.1111/1467-8659.00669.

[53] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, “Multi-view Convolutional Neural Networks for 3D Shape Recognition,” May 2015, doi: 10.1109/ICCV.2015.114.

[54] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, “Gradient-based learning applied to document recognition,” *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998, doi: 10.1109/5.726791.

[55] J. Deng, W. Dong, R. Socher, L. Li, Kai Li, and Li Fei-Fei, “ImageNet: A large-scale hierarchical image database,” in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2009, pp. 248–255, doi: 10.1109/CVPR.2009.5206848.

[56] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation,” in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 580–587, doi: 10.1109/CVPR.2014.81.

[57] C. V. Dung and L. D. Anh, “Autonomous concrete crack detection using deep fully convolutional neural network,” *Automation in Construction*, vol. 99, pp. 52–58, 2019, doi: https://doi.org/10.1016/j.autcon.2018.11.028.

[58] B. Zhong, X. Xing, P. Love, X. Wang, and H. Luo, “Convolutional neural network: Deep learning-based classification of building quality problems,” *Advanced Engineering Informatics*, vol. 40, pp. 46–57, 2019, doi: https://doi.org/10.1016/j.aei.2019.02.009.

[59] J.-C. Su, M. Gadelha, R. Wang, and S. Maji, *A Deeper Look at 3D Shape Classifiers*. 2018.

[60] T. Munakata, *Fundamentals of the New Artificial Intelligence: Neural, Evolutionary, Fuzzy and More*, 2nd ed. London: Springer-Verlag, 2008.

[61] A. Prieto, M. Atencia, and F. Sandoval, “Advances in artificial neural networks and machine learning,” *Neurocomputing*, vol. 121, pp. 1–4, 2013, doi: https://doi.org/10.1016/j.neucom.2013.01.008.

[62] D. Floreano, P. Dürr, and C. Mattiussi, “Neuroevolution: from architectures to learning,” *Evolutionary Intelligence*, vol. 1, no. 1, pp. 47–62, 2008, doi: 10.1007/s12065-007-0002-4.

[63] B. Kröse, B. Krose, P. van der Smagt, and P. Smagt, “An introduction to neural networks,” *J Comput Sci*, vol. 48, Jan. 1993.

[64] Rumelhart, D. E., J. Mcclelland, and J. L., *Parallel distributed processing: explorations in the microstructure of cognition. Volume 1. Foundations*. 1986.

[65] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. The MIT Press, 2016.

[66] M. Đalto, “Deep neural networks for time series prediction with applications in ultra-short-term wind forecasting,” *IEEE*, pp. 1657–1663, Feb. 2014.

[67] A. Ferreira and G. Giraldi, “Convolutional Neural Network approaches to granite tiles classification,” *Expert Systems with Applications*, vol. 84, pp. 1–11, 2017, doi: https://doi.org/10.1016/j.eswa.2017.04.053.

[68] Y. Bengio, “Learning Deep Architectures for AI,” *Foundations and Trends® in Machine Learning*, vol. 2, no. 1, pp. 1–127, 2009, doi: 10.1561/2200000006.

[69] I. Namatēvs, “Deep Convolutional Neural Networks: Structure, Feature Extraction and Training,” *Information Technology and Management Science*, vol. 20, Dec. 2017, doi: 10.1515/itms-2017-0007.

[70] S. Zhong, J. Wu, Y. Zhu, P. Liu, J. Jiang, and Y. Liu, “Visual Orientation Inhomogeneity Based Convolutional Neural Networks,” in *2016 IEEE 28th International Conference on Tools with Artificial Intelligence (ICTAI)*, 2016, pp. 477–484, doi: 10.1109/ICTAI.2016.0079.

[71] A. Krizhevsky, I. Sutskever, and G. Hinton, “ImageNet Classification with Deep Convolutional Neural Networks,” *Neural Information Processing Systems*, vol. 25, Jan. 2012, doi: 10.1145/3065386.

[72] S. Albelwi and A. Mahmood, “A Framework for Designing the Architectures of Deep Convolutional Neural Networks,” *Entropy*, vol. 19, no. 6, p. 242, 2017, doi: https://doi.org/10.3390/e19060242.

[73] S. Krig, *Computer Vision Metrics. Survey, Taxonomy and Analysis of Computer Vision, Visual Neuroscience, and Deep Learning*. Springer, 2016.

[74] J. S. Ren, W. Wang, J. Wang, and S. Liao, “An unsupervised feature learning approach to improve automatic incident detection,” in *2012 15th International IEEE Conference on Intelligent Transportation Systems*, 2012, pp. 172–177, doi: 10.1109/ITSC.2012.6338621.

[75] C. Affonso, A. L. D. Rossi, F. H. A. Vieira, and A. C. P. de L. F. de Carvalho, “Deep learning for biological image classification,” *Expert Systems with Applications*, vol. 85, pp. 114–122, 2017, doi: https://doi.org/10.1016/j.eswa.2017.05.039.

[76] T. Chen, R. Xu, Y. He, and X. Wang, “A Gloss Composition and Context Clustering Based Distributed Word Sense Representation Model,” *Entropy*, vol. 17, pp. 6007–6024, Sep. 2015, doi: 10.3390/e17096007.

Appendix A – Sample MATLAB Code for Generating Dataset

```
For i=1:1000
        subplot(2,2,1)
        rectangle('Position',[1 2 2 10])
        set(gca,'visible','off')

        subplot(2,2,2)
        rectangle('Position',[1 2 2 10])
        set(gca,'visible','off')


        subplot(2,2,3)
        pos = [2 4 2 2];
        rectangle('Position',pos,'Curvature',[1 1])
        axis equal
        set(gca,'visible','off')


        subplot(2,2,4)
        pos = [2 4 2 2];
        rectangle('Position',pos,'Curvature',[1 1])
        axis equal
        set(gca,'visible','off')
end
```

Appendix B – Developed CNN in Keras

```
import keras

from keras.models import Sequential

from keras.layers import Dense, Dropout, Flatten

from keras.layers import Conv2D, MaxPooling2D

from keras.utils import to_categorical

from keras.preprocessing import image

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

from keras.utils import to_categorical

from tqdm import tqdm

train = pd.read_csv('train.csv')

train_image = []

for i in tqdm(range(train.shape[0])):

    img = image.load_img('train/'+train['id'][i].astype('str')+'.png', target_size=(28,28,1), grayscale=True)

    img = image.img_to_array(img)

    img = img/255

    train_image.append(img)

X = np.array(train_image)
```

```
y=train['label'].values

y = to_categorical(y)

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=42, test_size=0.2)

model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=(64,64,1)))

model.add(Conv2D(64, (3, 3), activation='relu'))

model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Dropout(0.25))

model.add(Flatten())

model.add(Dense(128, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(10, activation='softmax'))

model.compile(loss='categorical_crossentropy',optimizer='Adam',metrics=['accuracy'])

model.fit(X_train, y_train, epochs=10, validation_data=(X_test, y_test))

test = pd.read_csv('test.csv')

test_image = []

for i in tqdm(range(test.shape[0])):

    img = image.load_img('test/'+test['id'][i].astype('str')+'.png', target_size=(64,64,1), grayscale=True)

    img = image.img_to_array(img)

    img = img/255

    test_image.append(img)
```

```
test = np.array(test_image)

prediction = model.predict_classes(test)
```

Appendix C – Naviswork Plugin Code

```
using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Autodesk.Navisworks.Api.Plugins;

using System.Windows.Forms;

using AddinRibbon.Ctr;


namespace NWPlugin

{


   [Plugin("AddinRibbon","Sina", DisplayName ="AddinRibbon")]

   [RibbonLayout("AddinRibbon.xaml")]

   [RibbonTab("ID_CustomTab_1", DisplayName ="Ribbon1")]

   [Command("ID_Button_1",Icon = "1-16.png",LargeIcon = "1-32.png", ToolTip ="Show a Message")]


   public class ClAddin : CommandHandlerPlugin

   {
```

```
public override int ExecuteCommand(string name, params string[] parameters)

{


   switch(name)

   {

      case "ID_Button_1":


         if(!Autodesk.Navisworks.Api.Application.IsAutomated)

         {


            var pluginRecord =
Autodesk.Navisworks.Api.Application.Plugins.FindPlugin("ClDockPanelUpdate.Sina");


            if(pluginRecord is DockPanePluginRecord && pluginRecord.IsEnabled)

            {


               var docPanel = (DockPanePlugin)(pluginRecord.LoadedPlugin ??
pluginRecord.LoadPlugin());


               docPanel.ActivatePane();

            }
```

```
                }

                break;

            }

            return 0;
        }
    }
}


namespace AddinDockPanel
{
    [Plugin("ClDockPanelUpdate", "Sina", DisplayName = "Dynamic Update")]
    [DockPanePlugin(200, 400, AutoScroll =true, MinimumHeight =100,MinimumWidth =200)]

    public class ClDockPanelUpdate: DockPanePlugin
    {
        public override Control CreateControlPane()
```

```
{
    var tc = new TabControl();
    tc.ParentChanged += SetDockStyle;

    var tp1 = new TabPage("Auto Update");
    tp1.Controls.Add(new UcUpdate());
    tc.TabPages.Add(tp1);

    var tp2 = new TabPage("Properties");
    tp2.Controls.Add(new UcProperties());
    tc.TabPages.Add(tp2);

    var tp3 = new TabPage("UcTools");
    tp3.Controls.Add(new UcTools());
    tc.TabPages.Add(tp3);

    return tc;

}
```

```
private void SetDockStyle(object sender, EventArgs e)
{
    try
    {
        var tc = sender as TabControl;
        tc.Dock = DockStyle.Fill;
    }

    catch (Exception)
    {

    }
}

public override void DestroyControlPane(Control pane)
{
    try
    {
        var ctr = (UcUpdate)pane;
        ctr?.Dispose();
    }
```

```
        catch (Exception)
        {
         //
        }


    }
  }


}
```

Appendix D – Naviswork Plugin Code for Viewpoint Generation

```
using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Drawing;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using Autodesk.Navisworks.Api;

using NavisworksApp = Autodesk.Navisworks.Api.Application;

using Autodesk.Navisworks.Api.ComApi;

using Autodesk.Navisworks.Api.Interop.ComApi;

using eColor = Autodesk.Navisworks.Api.Color;

using System.IO;

using Graphics = System.Drawing.Graphics;

namespace AddinRibbon.Ctr

{

    public partial class UcTools : UserControl

    {
```

```
public string LastSelName { get; private set; }

public string LastSelName2 { get; private set; }

public string LastSelName3 { get; private set; }

public string LastSelName4 { get; private set; }


public IEnumerable<ModelItem> LastSelection { get; private set; }

public IEnumerable<ModelItem> LastSelection2 { get; private set; }

public IEnumerable<ModelItem> LastSelection3 { get; private set; }

public IEnumerable<ModelItem> LastSelection4 { get; private set; }


public UcTools()

{

    InitializeComponent();

}


private void LbIsolate_MouseUp(object sender, MouseEventArgs e)

{

    IsolateSelection();

}


private void IsolateSelection()
```

```
{
    var acd = NavisworksApp.ActiveDocument;

    try
    {

        ModelItemCollection oModelColl =
Autodesk.Navisworks.Api.Application.ActiveDocument.CurrentSelection.SelectedItems;

        Autodesk.Navisworks.Api.Interop.ComApi.InwOpState oState5 = ComApiBridge.State;
        Autodesk.Navisworks.Api.Interop.ComApi.InwOpSelection oSel =
ComApiBridge.ToInwOpSelection(oModelColl);

        int k1 = 1;
        int k2 = 1;
        foreach (var item in NavisworksApp.ActiveDocument.CurrentSelection.SelectedItems)
        {
            ModelItemCollection hidden = new ModelItemCollection();

            ModelItemCollection visible = new ModelItemCollection();
```

```
if (item.AncestorsAndSelf != null)

    visible.AddRange(item.AncestorsAndSelf);

if (item.Descendants != null)

    visible.AddRange(item.Descendants);



foreach (ModelItem toShow in visible)

{

    if (toShow.Parent != null)

    {

        hidden.AddRange(toShow.Parent.Children);

    }

}
```

```
foreach (ModelItem toShow in visible)

{

    hidden.Remove(toShow);

}

Autodesk.Navisworks.Api.Application.ActiveDocument.Models.SetHidden(hidden, true);

((Autodesk.Navisworks.Api.Interop.LcOwViewer)acd.ActiveView.Viewer).LookFrom(Autodesk.Navisworks
.Api.Interop.LcOaPartitionViewDirection.eFRONT);

    var state = ComApiBridge.State;

    var cv = state.CurrentView.Copy();
```

```
var vp = state.ObjectFactory(nwEObjectType.eObjectType_nwOpView);

var view = vp as InwOpView;

view.ApplyHideAttribs = true;

view.ApplyMaterialAttribs = true;

vp.Name = item.DisplayName;

vp.anonview = cv;

state.SavedViews().Add(vp);

InwOpState10 oState = ComApiBridge.State;

InwOaPropertyVec options = oState.GetIOPluginOptions("lcodpimage");

foreach (InwOaProperty opt in options.Properties())

{

   if (opt.name == "export.image.format")

      opt.value = "lcodpexpng";
```

```
}

string snapshot = "C:\\Users\\sinaa\\Desktop\\C#\\Screenshots\\" + k1.ToString() + ".PNG";


oState.DriveIOPlugin("lcodpimage", snapshot, options);

System.Drawing.Bitmap oBitmap = new System.Drawing.Bitmap(snapshot);

System.IO.MemoryStream ImageStream = new System.IO.MemoryStream();

oBitmap.Save(ImageStream, System.Drawing.Imaging.ImageFormat.Jpeg);



k1 += 1;



((Autodesk.Navisworks.Api.Interop.LcOwViewer)acd.ActiveView.Viewer).LookFrom(Autodesk.Navisworks
.Api.Interop.LcOaPartitionViewDirection.eRIGHT);



var cv2 = state.CurrentView.Copy();


var vp2 = state.ObjectFactory(nwEObjectType.eObjectType_nwOpView);


var view2 = vp2 as InwOpView;
```

```
view2.ApplyHideAttribs = true;

view2.ApplyMaterialAttribs = true;

vp2.Name = item.DisplayName;

vp2.anonview = cv2;
state.SavedViews().Add(vp2);

InwOpState10 oState2 = ComApiBridge.State;

InwOaPropertyVec options2 = oState2.GetIOPluginOptions("lcodpimage");

foreach (InwOaProperty opt2 in options2.Properties())
{
   if (opt2.name == "export.image.format")
      opt2.value = "lcodpexpng";
}
string snapshot2 = "C:\\Users\\sinaa\\Desktop\\C#\\Screenshots\\" + k1.ToString() +
".PNG";

oState2.DriveIOPlugin("lcodpimage", snapshot2, options2);
System.Drawing.Bitmap oBitmap2 = new System.Drawing.Bitmap(snapshot2);
```

```
System.IO.MemoryStream ImageStream2 = new System.IO.MemoryStream();

oBitmap2.Save(ImageStream2, System.Drawing.Imaging.ImageFormat.Jpeg);


k1 += 1;


((Autodesk.Navisworks.Api.Interop.LcOwViewer)acd.ActiveView.Viewer).LookFrom(Autodesk.Navisworks
.Api.Interop.LcOaPartitionViewDirection.eBACK);


var cv3 = state.CurrentView.Copy();


var vp3 = state.ObjectFactory(nwEObjectType.eObjectType_nwOpView);


var view3 = vp3 as InwOpView;


view3.ApplyHideAttribs = true;


view3.ApplyMaterialAttribs = true;


vp3.Name = item.DisplayName;


vp3.anonview = cv3;
```

```
state.SavedViews().Add(vp3);


InwOpState10 oState3 = ComApiBridge.State;


InwOaPropertyVec options3 = oState3.GetIOPluginOptions("lcodpimage");


foreach (InwOaProperty opt3 in options3.Properties())
{
    if (opt3.name == "export.image.format")
        opt3.value = "lcodpexpng";
}
string snapshot3 = "C:\\Users\\sinaa\\Desktop\\C#\\Screenshots\\" + k1.ToString() +
".PNG";


oState3.DriveIOPlugin("lcodpimage", snapshot3, options3);
System.Drawing.Bitmap oBitmap3 = new System.Drawing.Bitmap(snapshot3);
System.IO.MemoryStream ImageStream3 = new System.IO.MemoryStream();
oBitmap3.Save(ImageStream3, System.Drawing.Imaging.ImageFormat.Jpeg);



k1 += 1;
```

((Autodesk.Navisworks.Api.Interop.LcOwViewer)acd.ActiveView.Viewer).LookFrom(Autodesk.Navisworks.Api.Interop.LcOaPartitionViewDirection.eTOP);

var cv4 = state.CurrentView.Copy();

var vp4 = state.ObjectFactory(nwEObjectType.eObjectType_nwOpView);

var view4 = vp4 as InwOpView;

view4.ApplyHideAttribs = true;

view4.ApplyMaterialAttribs = true;

vp4.Name = item.DisplayName;

vp4.anonview = cv4;

state.SavedViews().Add(vp4);

InwOpState10 oState4 = ComApiBridge.State;

InwOaPropertyVec options4 = oState4.GetIOPluginOptions("lcodpimage");

```
                foreach (InwOaProperty opt4 in options4.Properties())
                {
                   if (opt4.name == "export.image.format")
                       opt4.value = "lcodpexpng";
                }
                string snapshot4 = "C:\\Users\\sinaa\\Desktop\\C#\\Screenshots\\" + k1.ToString() +
".PNG";

                oState4.DriveIOPlugin("lcodpimage", snapshot4, options4);
                System.Drawing.Bitmap oBitmap4 = new System.Drawing.Bitmap(snapshot4);
                System.IO.MemoryStream ImageStream4 = new System.IO.MemoryStream();
                oBitmap4.Save(ImageStream4, System.Drawing.Imaging.ImageFormat.Jpeg);

                k1 += 1;


                String jpg1 = snapshot;
                String jpg2 = snapshot2;
                String jpg3 = snapshot3;
                String jpg4 = snapshot4;
                String jpg5 = "C:\\Users\\sinaa\\Desktop\\C#\\TF\\TF\\TF\\assets\\inputs-predict\\data\\" +
k2.ToString() + ".PNG";
```

```
String jpg6 =
"C:\\Users\\sinaa\\Desktop\\C#\\TF\\TF\\TF\\bin\\Debug\\netcoreapp2.1\\assets\\inputs-predict\\data\\" +
k2.ToString() + ".PNG";



Image img1 = Image.FromFile(jpg1);

Image img2 = Image.FromFile(jpg2);

Image img3 = Image.FromFile(jpg3);

Image img4 = Image.FromFile(jpg4);


int width = img1.Width + img2.Width + 50;

int height = img1.Height + img2.Height + 50;


Bitmap img5 = new Bitmap(width, height);

Graphics g = Graphics.FromImage(img5);


g.Clear(System.Drawing.Color.Black);

g.DrawImage(img1, new Point(0, 0));

g.DrawImage(img2, new Point(img1.Width + 50, 0));

g.DrawImage(img3, new Point(0, img1.Height + 50));

g.DrawImage(img4, new Point(img1.Width + 50, img1.Height + 50));
```

```
Bitmap img6 = new Bitmap(width, height);

Graphics g2 = Graphics.FromImage(img6);


g2.Clear(System.Drawing.Color.Black);

g2.DrawImage(img1, new Point(0, 0));

g2.DrawImage(img2, new Point(img1.Width + 50, 0));

g2.DrawImage(img3, new Point(0, img1.Height + 50));

g2.DrawImage(img4, new Point(img1.Width + 50, img1.Height + 50));


g.Dispose();

g2.Dispose();

img1.Dispose();

img2.Dispose();

img3.Dispose();

img4.Dispose();


img6.Save(jpg6, System.Drawing.Imaging.ImageFormat.Jpeg);

img5.Save(jpg5, System.Drawing.Imaging.ImageFormat.Jpeg);

img5.Dispose();
```

```
                AddRecord(k2.ToString() + ".png",
"C:\\Users\\sinaa\\Desktop\\C#\\TF\\TF\\TF\\bin\\Debug\\netcoreapp2.1\\assets\\inputs-
predict\\data\\image_list.txt");

                AddRecord(k2.ToString() + ".png",
"C:\\Users\\sinaa\\Desktop\\C#\\TF\\TF\\TF\\assets\\inputs-predict\\data\\image_list.txt");



                k1 += 1;

                k2 += 1;

                Autodesk.Navisworks.Api.Application.ActiveDocument.Models.SetHidden(hidden, false);



            }



        }



        catch (Exception)

        {

            //

        }

    }



    public static void AddRecord(string name, string filepath)

    {
```

```
    using (System.IO.StreamWriter file = new System.IO.StreamWriter(@filepath, true))
    {
        file.WriteLine(name);
    }

}


private void LbSave_MouseUp(object sender, MouseEventArgs e)
    {
         SaveCurrentViewpoint();
    }

private void SaveCurrentViewpoint()
{

}

private void LbIsolateSave_MouseUp(object sender, MouseEventArgs e)
{
    try
    {
```

```
            IsolateSelection();
        }
        catch (Exception)
        {
            //
        }
    }
    protected override void OnParentChanged(EventArgs e)
    {
        base.OnParentChanged(e);
        Dock = DockStyle.Fill;
    }
  }
}
```