Journal of Building Engineering 78 (2023) 107672

Contents lists available at ScienceDirect

Journal of Building Engineering

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

Full Length Article

Generating BIM model from structural and architectural plans using Artificial Intelligence[☆]

Martin Urbieta [a,b], Matias Urbieta [a,b,*], Tomas Laborde [a], Guillermo Villarreal [a], Gustavo Rossi [a,b]

[a] Centro de investigación LIFIA, Facultad de Informática, Universidad Nacional de La Plata, Calle 50 y 120, La Plata, 1900, Buenos Aires, Argentina

[b] Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET), Godoy Cruz 2290, Ciudad Autonoma de Buenos Aires, C1425FQB, Buenos Aires, Argentina

ARTICLE INFO

Keywords:
BIM
Machine-learning
Blueprints
Floor plans
As-build
Building
Model generation
IFC
2D drawings
Architectural plans
Structural plans

ABSTRACT

Over the last few decades, building development has been recorded using hand-made blueprints before CAD tools appeared and later with digital building plans. As a consequence, there is a large amount of information in millions of assets that are hard to process because of their analog nature. Since adopting the Building Information Model (BIM) approach, any new building plan can be subject to sophisticated validations and analysis. However, legacy analog plans cannot profit from sophisticated BIM analysis, and it is not feasible to manually generate BIM representations at low cost. There is a demand for BIM models of existing buildings that are feasible to be integrated into a workflow for building energy retrofitting. This paper presents a novel approach to generating BIM Models based on artificial intelligence algorithms by parsing architectural and structural drawings. To identify elements from blueprints and generate the model, we first trained the Mask R-CNN framework with our dataset of 9 concrete buildings composed of architectural and structural blueprints. The outcome of the process is a BIM model corresponding to one of the multi-storey buildings using the Industry Foundation Classes (IFC) format. Building development has been recorded using hand-made blueprints before CAD tools appeared and later with digital building plans.

1. Introduction

Building Information Modeling (BIM) is defined by the International Organization for Standardization (ISO) as “the use of a shared digital representation of a built object (including buildings, bridges, roads, process plants, etc.) to facilitate design, construction and operation processes to form a reliable basis for decisions”[1]. The acronym BIM also stands for “the shared digital representation of the physical and functional characteristics of any construction works” [1]. In this work, we adhere to the second definition because the final result of our work is an exchangeable model between different BIM tools. Building Information Modeling is a rising methodology in the construction industry. It is used for the creation and management of digital information assets (i.e. digital twins) throughout its entire life cycle, from planning and design to construction and operations. It represents an opportunity to improve the user’s well-being, energy efficiency, flexibility, and resilience of existing buildings.

☆ Acknowledgments - The research work was supported by the Consejo Nacional de Investigaciones Científicas y Técnicas (CONICET).

* Corresponding author at: Centro de investigación LIFIA, Facultad de Informática, Universidad Nacional de La Plata, Calle 50 y 120, La Plata, 1900, Buenos Aires, Argentina.

E-mail address: matias.urbieta@lifia.info.unlp.edu.ar (M. Urbieta).

https://doi.org/10.1016/j.jobe.2023.107672
Received 29 April 2023; Received in revised form 24 August 2023; Accepted 27 August 2023
Available online 6 September 2023

The Industry Foundation Classes (IFC) is a CAD data exchange data schema for BIM-based data exchange and interoperability defined by buildingSMART International (bSI)[1] as “a standardized, digital description of the built environment, including building and civil infrastructure. It is an open, international standard [2], meant to be vendor-neutral or agnostic, and usable across a wide range of hardware devices, software platforms, and interfaces for many different use cases”. IFC is an object-oriented schema that codifies the identity and semantics of elements (i.e. beams or slabs), attributes (i.e. material), and relationships of objects, abstract concepts, processes, and people. Because of the high-level building model, it is replacing the previous computer aid design (CAD) based on vector representation where there is no semantic information about plan elements (i.e. wall, door, beam, etc.) and their relationships (i.e. what door is embedded in a wall).

Worldwide BIM adoption has been boosted due to its benefits after implementation in new constructions [3] reducing errors and cost savings. The Return On the Investment (ROI) on case studies varied greatly from 16% to 1.654% [4]. The cost in the engineering, construction, and operation phases, due to implementing BIM in new building processes, falls by 15%–25% [5]. Even so, there is no real need to create as-built BIM for existing buildings [6], but there is a technical need for BIM in energy retrofitting projects for the pre-energy modeling stage [7]. If BIM as-built models of existing constructions are available, it can be effectively incorporated into a workflow for building retrofit [6]. An approach that reduces the cost of implementation by using automation will increase the benefits of the BIM approach.

In nearly every city, a public office is responsible for issuing Building Permits (BP) and maintaining the building file. Over time, the process of obtaining building permits has evolved from traditional paper-based processes to 2D digital data-based processes, and now to fully digitized settings based on BIM and GIS [8,9]. This BIM-model requirement for a Digital Building Permit (DBP) – to be issued by a county inspector – implies that a new building could be subject to new and more sophisticated validations and analyses. Although there are valuable assets in printed plans or digital vector/image files, the lack of a high-level model like BIM presents limitations in their usage for sophisticated analysis like retrofitting projects, City Information Model management, big data analysis, machine learning applications based on BIM and others.

Unfortunately, BIMs as-built for existing buildings are not available because Digital Building Permit requirements only apply to new projects including new buildings or retrofitting.

BIM models are valuable data sources to establish the City Information Model (CIM) [10]. The BIM adoption could ease the CIM development by aggregating new buildings that are designed with the BIM process. However, when dealing with existing buildings, generating models often require on-site surveys or utilizing as-built documentation. The proposal of Kippers et al. [11] presents a methodology to integrate external information from models generated from Light Detection And Ranging (LiDAR) flights over Amsterdam, available on the 3D BAG platform [12], with the internal information extracted from archived architectural plans, using machine learning methods. Retrofitting a building involves changing its systems or structure after its construction and occupation. With advances in technology, building retrofits can significantly improve sustainability. For energy retrofitting methodologies that require as-built BIM, new developments help to create the 3D building model to assess as-built conditions by measuring and acquiring on-site geometric data [13]. BIM As built are used as input in novels workflows for optimizing envelope design, energy efficiency, [14] and building retrofits [6] among others. BIM models could also feed a Machine learning solution that helps with the selection of the best retrofit opportunities to improve the green building system [15] but also with a design assistant implementation for AEC professionals. The raising of artificial intelligence tools aiding software developers like Co-pilot[2] can inspire approaches to aid AEC project designs that rely on big data sourced on building models. Translating legacy blueprint plans into an object-oriented model like BIM is a key part of this sort of solution.

The availability of BIM models at low cost for existing buildings, generated automatically from archived architectural plans, can provide valuable information to stakeholders for decision-making regarding whether or not to proceed with building retrofits, based on Data Science on BIM models dataset. Manual translation from plans to models is error-prone and time-consuming if we consider what makes its massive processing unfeasible in the case of municipalities or cities. These existing as-built survey technologies, including laser scanning, photogrammetry, 3D camera ranging, topographic methods, and videogrammetry, share common drawbacks: expensive and fragile equipment, the need for trained operators, and potential time-consuming processes.

However, Machine Learning (ML) algorithms have the potential to significantly improve this process by enabling the translation of plans into models within seconds, benefiting both the academia and industry by allowing for big-data analysis, validation models, and more. The challenge of creating models of actual buildings from 2D scanned plans or CAD digital drawings has been approached with various techniques. This research topic mostly focuses on floor plans while other types of drawings are ignored [16]. The complexity of tasks involved in recognizing a building’s architectural features and its structural, mechanical, electrical, and plumbing systems requires further research to develop methods for modeling entire buildings from existing building records.

Machine learning could be used to support as-built BIM model generation automatically. Still, there are some barriers in the building industry, for example, the lack of large-scale labeled datasets to train and validate the model so that it can be used on different types of buildings, rather than being restricted to specific ones [15]. The quality and size of the dataset used for training directly impacts the quality of outcomes achieved by machine learning solutions., but there is a lack of datasets for real as-built constructions. Available floor plan datasets [17] are mainly focused on architectural features, which do not allow to get a full building representation. Therefore, structural, mechanical, electrical, and plumbing blueprints are required to generate a holistic model for a building.

1 https://technical.buildingsmart.org/standards/ifc/ Accessed on 7th November 2022
2 https://copilot.github.com/

To make matters worse, there is no standard plan element drawing style. Thus, there is a wide variety of ways for representing elements in plans, which depend on local styles – which are continuously updated – and designers' (i.e., architects and engineers) visual preferences. As a consequence, a trained algorithm based on blueprints honoring local drawing standards may not have a good performance when classifying plans compliant with a different drawing style or standard. There is an opportunity to generate rich documents for modeling buildings from building documentation available at Municipal/County authorities who requested the plans as part of required documentation when issuing the building permits (BP). There are decades of architecture, engineering, and construction (AEC) industry plans archived and accumulated that meet the local quality standards and can be used as a basis for creating datasets to be processed by machine learning algorithms.

This paper presents a novel approach to generating BIM Model based on artificial intelligence algorithms.

The contribution of this work is many-fold:

- An approach for identifying drawing elements from plans using machine learning.
- Novel public[3] dataset with structural and architectural plans .
- Building Model Information model generation combining elements extracted from structural and architectural plans.
- The approach is extensible to include AEC specialties like mechanical, electrical, and plumbing plans.
- We illustrate our approach on a building case study.

BIM model generation uses the Python library IfcOpenShell[4] producing interchangeable documents. From now on, we will refer to BIM model and IFC file for representing the output of our process.

Related work is presented in Section 2. In Section 3, we introduce our approach. In Section 4, the approach is illustrated with a case study. Pros, Cons and Limitations of this work is presented in Section 5. And finally, the conclusions and further work are presented in Section 6.

2. Related work

For the research selection process of related works, we reviewed articles from the Scopus database and Google Scholar. The first one provides us with a solid academic resource, and the second one provides more gray literature results. We also investigated the references of the resulting set of publications — a process known as snowballing. Our main research questions during the related work review were:

- RQ1: What is the state of the art regarding deep learning in construction?
- RQ2: Are there any specialized datasets of construction blueprints published?
- RQ3: Is there any methodology for automatically generating BIM models from construction plans?
- RQ4: Which machine learning models have been used for BIM model generation?

Our four research questions contain the following keywords: "BIM model, Dataset, blueprints, machine learning".A list of synonyms was constructed for each of these words, as in the example for research question 3 which contains keywords 'BIM models', 'automatically', 'generation', 'construction plans':

```
((BIM model*) OR IFC OR (As-is BIM) OR (As-is IFC)) AND (automatic* OR semi*
    automatic*) AND (generation* OR 3D reconstruction* OR modeling* OR creation
    ) AND ((floor*plan*) OR blueprint* OR (2D plans*) OR (scanned plans) OR  (
    architectural drawings) OR (structural drawings) OR (existing buildings))
```

Our list of search terms was adapted to match each research question and the individual requirement of the search engines on our source list. We include journal articles, conference papers, and reviews. The search was applied to the full text and we limit to the papers written in English.

2.1. *Model generation from mapping 2D plans*

The search for a methodology to generate construction models from mapping plans is not new. Although there are methodologies for 3D modeling, the emergence of BIM, requires an object-oriented model with an interchangeable IFC format. Gimenez et al. [18] developed a C++ prototype, for generating IFC 3D model from 2D scanned plans, including walls, openings, and spaces. The main drawback is the need of reviewing the processing algorithm when implementing the approach on a location having a different drawing style as well as the need of programming a new algorithm for any new blueprint like Mechanical, Electrical, and Plumbing (MEP). An ML approach could avoid this issue by retraining the network with the new dataset (having a custom drawing style) and type of plan elements.

In the research presented by Zhu et al. [19] the authors reconstruct 3D buildings from 2D vector plans. First, structural components are recognized (SC) with a shape-opening graph (SOG) that describes the relationship in a hierarchy tree which enables them to be used in Revit; however, the height is set as default values for multi-floor reconstruction and scope to walls or openings.

[3] Datasets and scripts https://bitbucket.org/plans2bim/paper-struct-arq2bim/src/master/

[4] http://ifcopenshell.org/

A novel deep learning method for floor plan parsing was introduced by Liu et al. [20], in which a Convolutional Neural Network (CNN) first transforms a rasterized image into a set of junctions to represent low-level geometrics (i.e. wall points). These are later aggregated using Integer Programming (IP) into a set of primitives (i.e. wall lines). The result produces a vectorized floor plan, which allows a 3D model representation. However, the approach parses only architectural plans, ignoring wall thicknesses, limiting it to vertical and horizontal walls, and the 3d generated popup is not a BIM-IFC model.

2.2. BIM model generated from CAD drawings

There are techniques to generate BIM models from editable CAD files, processing the vector elements. These techniques are applicable to existing buildings that have CAD documents. The approach proposed by Yang et al. [21] relies on a Dynamo plug-in in Revit and it is applied in two case studies: an eight-storey concrete frame structure from an office building, including one roof and a two-storey steel frame structure that contains a standard-floor and a slope-roof-floor. The method requires processing the raw CAD drawing to classify the information into layers, which is carried out by a technician. As a result, the method generates a 3D structural BIM model from structural CAD plans with a semiautomatic method extracting elements information. In the same way, Bortoluzzi et al. [22] create a Facility Management (FM) BIM model with Computer Aid (CAFM) data, by pre-processing architectural DWG drawings and the floor elevation for creating models with façade, openings, rooms by using Dynamo and scripts in a variety of 17 building shapes with complex floor plans that belong to the Ryerson University Campus. The method requires intervention by a technician to reduce the raw CAD drawing up to three sets of data on a dedicated layer: room boundaries, exterior building boundaries, and room number tag. Yin et al. work [23] also succeeded in reconstructing a façade and interior walls into a 3D BIM model including exterior openings and extracting height information from elevation drawings. However, the approach cannot infer the internal building components' height which is a critical attribute used for element modeling like walls. In the same way, Lu et al. [24] propose a semi-automatic solution for 2D structural CAD drawings processing generating an IFC-based structural BIM object. A tool built in Matlab is described, including data extraction with Optical Character Recognition (OCR). In the research presented by Wen et al. [25], the authors describe an algorithm focused on walls, implemented by Autodesk Revit customization. It includes both geometrical and topological relations to provide a 3D building model; however, the algorithm does only support horizontal and vertical walls.

Unfortunately, CAD drawings files are not available publicly and they are only accessible by their owners; it is required to pre-process the file or set files, which normally are unstructured, may contain deprecated information and can require removing superfluous content. On the other hand, CAD files have the advantage of being able to identify the sizes of the represented elements, as long as they have not been scaled or decorated for presentation. The semi-automatic nature of these approaches requires human resources to get a BIM document. This is a challenge to bulk-process thousands of plans. Moreover, they require licenses for BIM tools to design BIM models. Additionally, CAD documents like DWG are not always available for existing buildings.

2.3. Machine learning applied on 2D plans

Zhao et al. [26] proposed the convolutional neural network called You Only Look Once (Yolo) [27] implemented in 2D scanned CAD structural drawings for object recognition of five classes of elements: grid, reference, column, horizontal beam, vertical beam, and sloped beam. The classified object is located by a bounding box. An image preprocessing that included gray processing, binarization and color inversion, and morphological operation, was applied before labeling and augmenting the images. The final output was a TXT file containing the classification and locations of the detected elements. The work does not discuss in depth how to capture element attributes like length, width, and height mandatory for creating any BIM models. The same authors [28] presented an implementation of faster region-based convolutional neural network [29,30] (Faster R-CNN), trained on a non-public dataset including 500 framing plans honoring the British Standard (BS 8110). The work concluded that Faster R-CNN is slightly better than YOLO after comparing their detection performance. The method presented a transformation from Pixel Coordinate System (PCS) to Drawing Coordinate System (DCS) and achieved to generate a BIM IFC model. Zeng et al. [31] presented a multi-task neural network recognizing room-boundary, room-type elements in architectural floor plans, introducing R2V and R3D datasets. In this case, there is no 3D or BIM model presented for a building case, just isolated plans. Seo et al. [32], due to the lack of a dataset, used Korea Land and Housing Corporation's House Floor Plans as the dataset, and implemented Google DeeplabV3+ framework for recognizing architectural floor plan elements and spaces without producing a BIM model.

Dodge et al. [33] introduced a new real estate floor plan dataset named R-FP, and proposed a fully convolutional network (FCN) with a stride of 2 for the recognition wall. By implementing OCR and object detection, they estimated the room sizes. The LIFULL HOME's dataset [20], which contains 5 million floor plan images, was selected for training though only 1000 of them were randomly sampled. There are other public datasets proposed for architectural floorplans, such as Rent3D [34], CubiCasa5k [35], CVC-FP [36].

The approach presented by Wu et al. [37] evaluates the framework Mask R-CNN for indoor mapping and modeling (IMM) parsing floor plans available on CVC-FP, simplifying the boundary elements into rectangular shapes and optimizing resolving topological conflicts between rectangles. However, the approach excludes curved walls and no BIM model is generated.

The Mask R-CNN framework presented by He et al. [38] is based on the Fast/Faster R-CNN and a Fully Convolutional Network (FCN) [39] framework enabling instance segmentation, which combines object detection, for classifying individual objects and localizes them using a bounding box, and semantic segmentation, for classifying each pixel into a fixed set of categories, differentiating objects instances. Instance segmentation can recognize complex figure shapes to be modeled in BIM. Mask R-CNN is an extension of Faster R-CNN added with Region of Interest Align (RoIAlign) and using the Feature Pyramid Network (FPN) proposed by Lin et al. [40].

2.4. Machine learning on BIM models

There are experiences where machine learning is also applied directly to BIM models, to cover modeling gaps, in specific analyses. In this context, Bloch et al. [41] applied machine learning for classifying room types in the context of residential apartments by running an Artificial Neural Network (ANN) on AZURE ML platform. Its training was based on a dataset of 32 BIM apartment models as input. In this case, the BIM models are enhanced with inferred information using the neural network. The proposed method was tested for semantic enrichment within the BIM model, to reduce incompatibilities such as code compliance when exporting the model to other software. Although this machine learning approach does not apply to the generation of the model itself, it does contribute to its improvement, with the potential to evaluate, for example, the impact of implementing a change in the regulation where room classification intervenes.

3. Proposed method

3.1. Goal and problem formulation

Generating enriched-data BIM models from 2D plans requires processing many different systems, documented using analog supports such as paper or 2D vectorized plans. The documentation may also include methods statements, diagrams, or even document revisions. The proposed methodology receives as input (but is not limited to), architectural and structural blueprints that are available as images (i.e. JPG/GIF) of the paper-based plans. These files can be used in different phases: first, digital assets are used for training the ML model, and in the second phase, models are the input for obtaining its BIM representation.

The steps to generate a single BIM model file from a complete set of construction drawings rely on a neural network trained individually on each AEC specialty, with the aim of extracting only the elements that belong to that category. As each specialty represents some specific system of the building, we understand that it must be mapped within as well, i.e., extracting column information from a plumbing blueprint could be an error, because the specialist is not supposed to design that element. This methodology requires having pre-classified each plan in architecture, structure, mechanics, plumbing, etc., to select the appropriate training for mapping.

Our approach aims to establish steps that can be fully automated to achieve bulk model generation throughout the mass plan processing. On the opposite side, there are proposals [21,22,24], with semi-automatic steps that limit complete modeling automation.

We chose to use, (but the approach is not limited to), the Mask R-CNN algorithm for training the ML model because it allows identifying irregular elements as masks. Many buildings have non-rectangular elements that are impossible to capture using rectangles like semi-circle shaped balconies, or sloped beams (which are not aligned to the X and Y axis). Another key factor is the extraction of the polygon's shape from the masks of the detected elements. Besides the shape simplification [37], this problem is also approached by Zorzi et al. [42], implementing a *fully convolutional network* (FCN) for corners detection.

Our literature review helped to confirm the diversity of drawing styles that makes it hard to design a solution that could be applied worldwide or even countrywide because not all locations share the same standards. No approach was able to provide a solution to this problem. As AEC designers' drawing styles, plan idioms, and local standards can produce a wide style diversity for representing elements and notations, this aspect should be considered to define the datasets for training and evaluation. Our approach faces this issue by training the system using drawings that honor local standards. Training only requires a technician to classify and tag images and then it can be triggered without any code change. Any code change is expensive because development tasks and different engineer roles are involved and it might be error-prone.

In Fig. 1, we introduce our approach that is many-fold:

- In the first place, there is a tagging and training step where local blueprint styles are captured and the main outcome of this step is a set of models where each one is focused on a specific specialty of AEC blueprints (e.g. a model will be focused on capturing beam, slab, and columns of the structure blueprint). A throughput description of this step is provided in Section 3.4.1.
- Some metadata information is extracted from the Title block element in the plan set. This would be used for identifying plans belonging to the same project when performing bulk processing of several buildings.
- Before processing the blueprints, the image size and scale are normalized across plans. Printed blueprints are required to include the element scales; however, during their digitization process, the blueprints might be scanned using different image resolutions making it hard to locale elements across blueprints in the same position. This problem has been discussed in depth on [16]. So, we apply feature alignment techniques [43] to align and resize different blueprints using shared elements, such as columns, spread on all blueprints. In Fig. 2(a) we show an architecture plan (Fig. 2) and a structure plan (Fig. 2(c)). The architecture plan is rotated in Fig. 2(b) and the final result after re-scaling images is shown in Fig. 2(d) where columns are aligned. This is mandatory to work with element location across maps to build a single BIM document.
  We decided to adjust images by generating temporal image versions so that analysts can visually debug any inconsistency with blueprints. After the BIM document is generated, temporal-scaled images are deleted. Another approach is to compute the offset and scale ratio and use them to transform elements when generating the BIM model. The main drawback is that any inconsistency will be hard to analyze visually because images will not be on the same scale.
- The next step is to classify blueprints and run the right ML model on them.

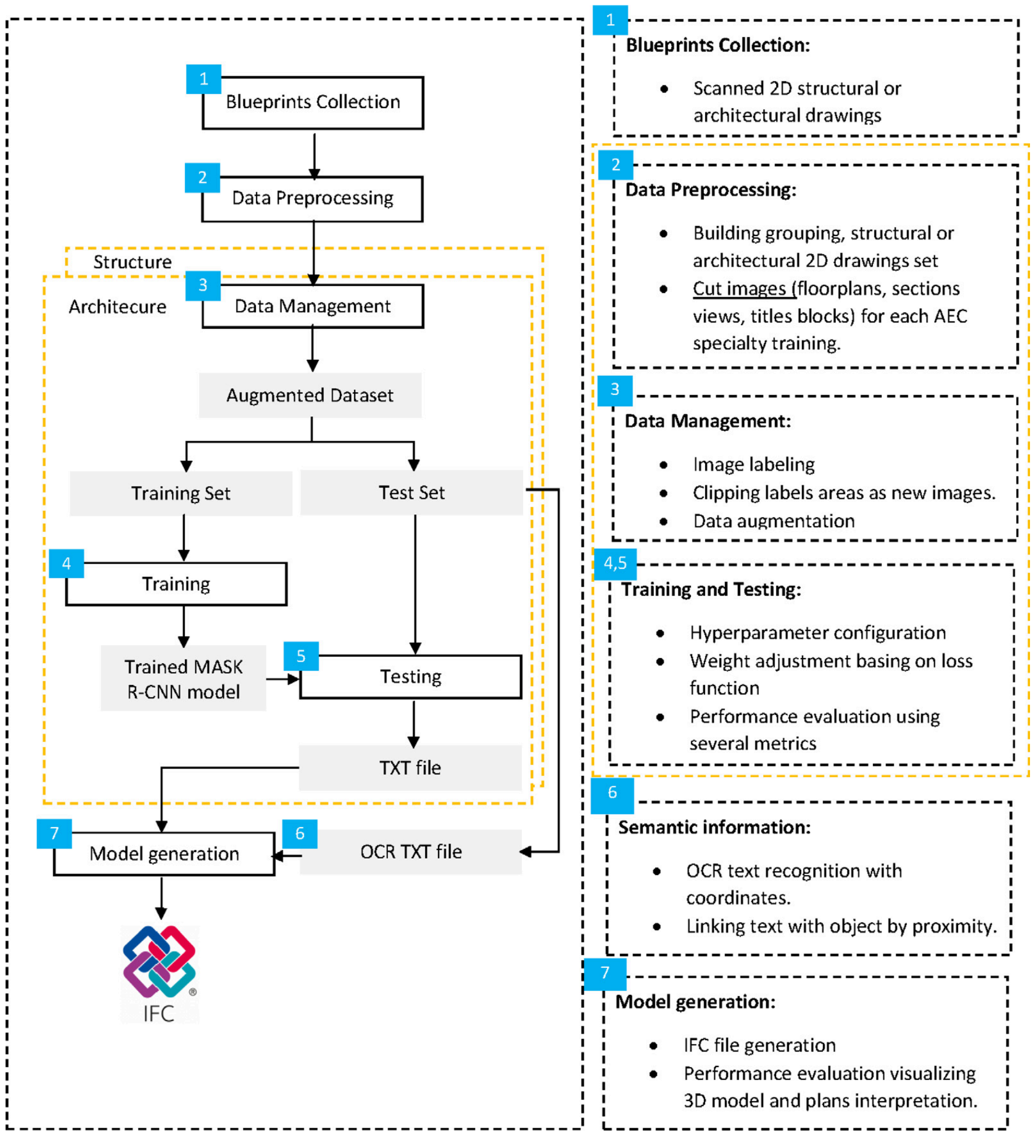
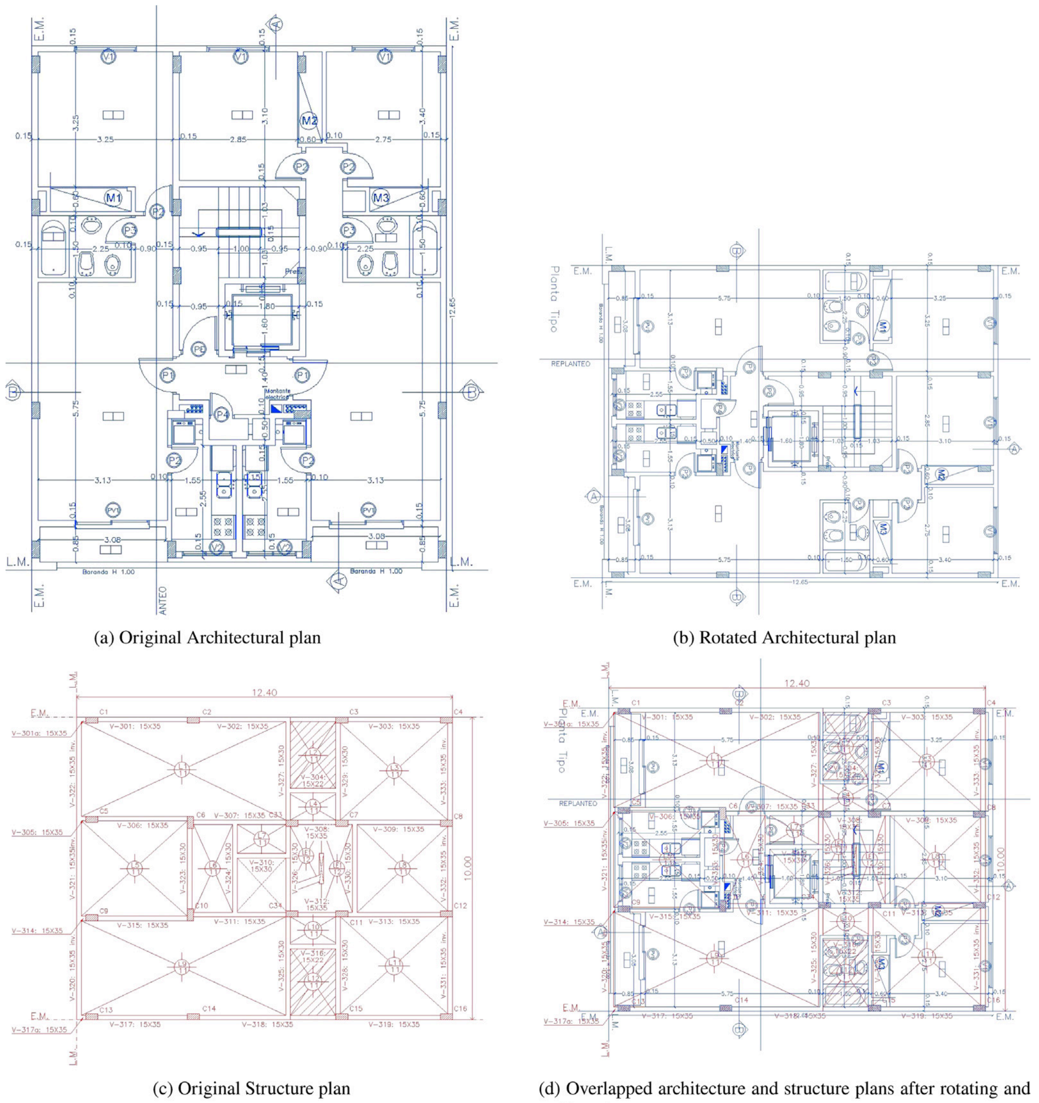

1 Blueprints Collection
2 Data Preprocessing
Structure
Architecure
3 Data Management
Augmented Dataset
Training Set
Test Set
4 Training
Trained MASK R-CNN model
5 Testing
TXT file
7 Model generation
6 OCR TXT file
IFC
1 Blueprints Collection:
Scanned 2D structural or architectural drawings
2 Data Preprocessing:
Building grouping, structural or architectural 2D drawings set
Cut images (floorplans, sections views, titles blocks) for each AEC specialty training.
3 Data Management:
Image labeling
Clipping labels areas as new images.
Data augmentation
4,5 Training and Testing:
Hyperparameter configuration
Weight adjustment basing on loss function
Performance evaluation using several metrics
6 Semantic information:
OCR text recognition with coordinates.
Linking text with object by proximity.
7 Model generation:
IFC file generation
Performance evaluation visualizing 3D model and plans interpretation.

Fig. 1. Overall approach schema.

- The AEC element classification process uses the trained models to get the elements and position information. Additionally, explicit and implicit information is extracted. On one hand, it considers the application of OCR to extract textual information like beam dimensions which is documented along with the bean drawing or the door/window id. Moreover, relationships between objects are identified such as the "host" relationship between the wall and door. This step will be described in Section 3.6.
- BIM generation is the final step where the elements and metadata are woven to compose a BIM model. This step will be described in Section 3.8.

As above mentioned, the proposed approach merges architectural, and structural plans in one building BIM model composed of many floors; they are obtained from 2D scanned plans, postponing a future work integration of plumbing, electricity, and mechanics plans. This can be easily extended to support more AEC specialties producing a richer BIM model. When BIM is used, technicians produce a shareable single model for new buildings that other professionals can incrementally enhance with additional details. Before the BIM age, a project could be spread into many individual plans which are views of the buildings without an underlying model. There are many printed or hand-made plans archived that can be targeted by our approach. In this work, we aim at combining different plans as sources for generating a single building model, which differs from the approaches introduced in the Related work section, which only focus on either Architectural or structural plans but not both.



(a) Original Architectural plan

(b) Rotated Architectural plan

(c) Original Structure plan

(d) Overlapped architecture and structure plans after rotating and scaling plans

Fig. 2. Plans transformation for aligning elements.

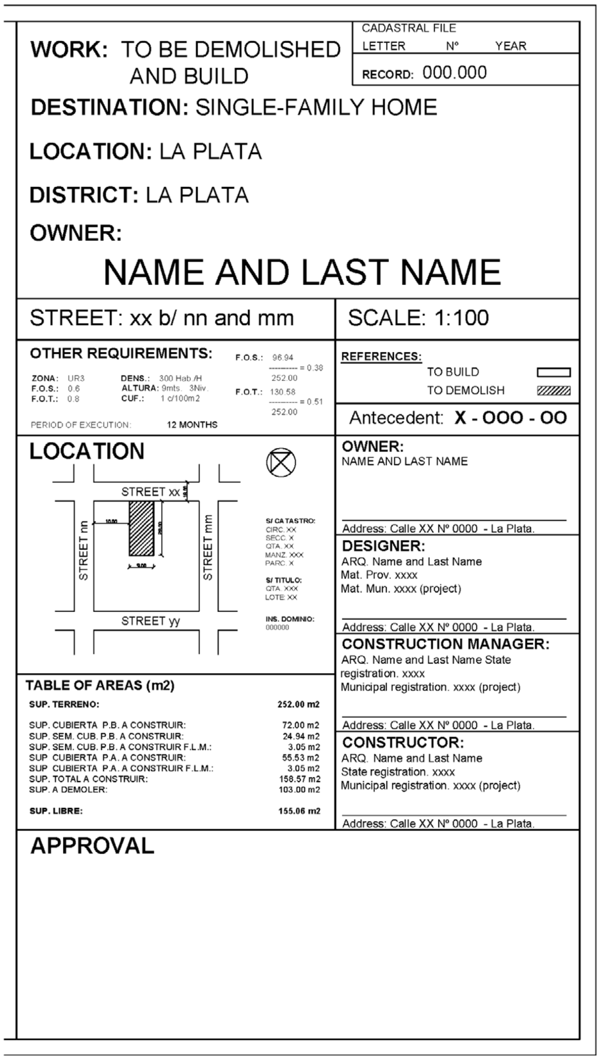
3.2. Blueprint collection

Buildings are often modeled using different drawing plans that are specialized in specific topics. For example, structure plans depict the main elements that give stability to the building. Moreover, each type of drawing relies on a specific set of valid elements that make sense in that context. For example, the structural plan combines beams, slabs, and columns among other elements. Finally, the elements might have some sort of textual annotation that is used to provide additional information like dimensions in the case of beams or to assign an identifier, which is used for referencing complementary information often provided in an attached document like doors and windows designs. In Table 1, we list the building plans and the elements considered in this work. For each element type, an element image, the required attributes for its modeling, the source of each attribute, and the IFC element used are provided. Finally, it is also documented whether or not the element has a textual or graphical annotation.

The raw blueprints had to be pre-processed and classified. First, they were grouped by building, from which architecture and a structure set were generated. The set was cut and separated by floor plan or section view, which were later labeled.

3.2.1. Title blocks for metadata project processing and classification

The title blocks contain relevant metadata required in BIM models such as Facility Address, Project Name, Description, Land title number, and address, among others.



WORK: TO BE DEMOLISHED AND BUILD

CADASTRAL FILE
LETTER N° YEAR
RECORD: 000.000

DESTINATION: SINGLE-FAMILY HOME

LOCATION: LA PLATA

DISTRICT: LA PLATA

OWNER:

NAME AND LAST NAME

STREET: xx b/ nn and mm

SCALE: 1:100

OTHER REQUIREMENTS:

ZONA: UR3
F.O.S.: 0.6
F.O.T.: 0.8

DENS.: 300 Hab./H
ALTURA: 9mts. 3Niv.
CUF.: 1 c/100m2

F.O.S.: 96.94 / 252.00 = 0.38

F.O.T.: 130.58 / 252.00 = 0.51

PERIOD OF EXECUTION: **12 MONTHS**

REFERENCES:
TO BUILD ▭
TO DEMOLISH ▨

Antecedent: **X - OOO - OO**

LOCATION

STREET xx
STREET nn
STREET mm
STREET yy
10.00
28.00
9.00

S/ CATASTRO:
CIRC. XX
SECC. X
QTA. XX
MANZ. XXX
PARC. X

S/ TITULO:
QTA. XXX
LOTE: XX

INS. DOMINIO:
000000

OWNER:
NAME AND LAST NAME

Address: Calle XX N° 0000 - La Plata.

DESIGNER:
ARQ. Name and Last Name
Mat. Prov. xxxx
Mat. Mun. xxxx (project)

Address: Calle XX N° 0000 - La Plata.

CONSTRUCTION MANAGER:
ARQ. Name and Last Name State registration. xxxx
Municipal registration. xxxx (project)

Address: Calle XX N° 0000 - La Plata.

CONSTRUCTOR:
ARQ. Name and Last Name
State registration. xxxx
Municipal registration. xxxx (project)

Address: Calle XX N° 0000 - La Plata.

TABLE OF AREAS (m2)

SUP. TERRENO:	**252.00 m2**
SUP. CUBIERTA P.B. A CONSTRUIR:	72.00 m2
SUP. SEM. CUB. P.B. A CONSTRUIR:	24.94 m2
SUP. SEM. CUB. P.B. A CONSTRUIR F.L.M.:	3.05 m2
SUP CUBIERTA P.A. A CONSTRUIR:	55.53 m2
SUP CUBIERTA P.A. A CONSTRUIR F.L.M.:	3.05 m2
SUP. TOTAL A CONSTRUIR:	158.57 m2
SUP. A DEMOLER:	103.00 m2
SUP. LIBRE:	**155.06 m2**

APPROVAL

Fig. 3. Title block.

A bulk plans mapping must be able to identify project information for classifying and grouping plans that belong to the same project facility. In Fig. 3, a standard title block is approved by the local Municipal authority for Building Permits (BP). The title blocks may vary according to local requirements; once defined, they become standard for applying a BP. The label is divided into compartments and specific information can be extracted from each one. Associating each keyword to objective metadata, through OCR techniques, the required information can be extracted.

3.2.2. Scope of elements to detect for modeling

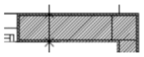
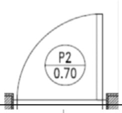
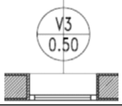
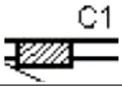
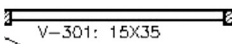
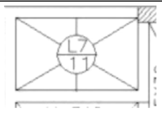
This work studies how to process the most important Architectural, Structural, and Section plans elements regarding the same building. In order to capture all the information available on plans, we combine object detection using machine learning, Optical Character Recognition, and scripting to construct IFC elements. As described in Table 1, we consider, but are not limited to, the following main elements in our approach that are present in plans like shown in Fig. 4.

Structural plans. Our structure plans database, as shown in Fig. 4(b), presents foundation elements as bases or piles, beams, columns, slabs, and stairs. Concrete steel bars reinforcement information are not included in these plans. For the sake of simplicity, foundation and stairs elements are kept out of scope. Stairs modeling is a challenge by itself; Lin [44] considered 54 different stair types for evaluating an automatic path generation on 3D building models in IFC format.

We considered the following main blueprint elements:

- **Columns.** Our scope is restricted to concrete rectangular sections. Section shape is obtained by the Cartesian's points from mapping 2D structure floor plans. The column's height is sourced from section view plans, but if they are not available, the

Table 1
Plan elements.

AEC Plan	Object	Image	Geometric information	Attribute source	Hosted by	IFC element	Textual/ Graphical Annotations
Architecture	Wall		Length Witdh	Architecture floor plan		IfcWallStandartCase	
			Height	Section view[a]			
Architecture	Door		Length Witdh	Architecture floor plan	Wall	IfcDoor	Door type reference
			Height	Carpentry detail[a]			
Architecture	Window		Length Witdh	Architecture floor plan	Wall	IfcWindow	Window type reference
			Height	Carpentry detail[a]			
Structure	Column		Length Witdh	Structure Plan		IfcColumn	Column id
			Height	Section view[a]			
Structure	Beam		Length Witdh	Structure Plan		IfcBeam	Beam ID, dimension annotation (i.e. height x width)
			Height	Section view or annotation[a]			
Structure	Slab		Length Witdh	Structure Plan		IfcSlab	
			Height	Section View or annotation[a]			

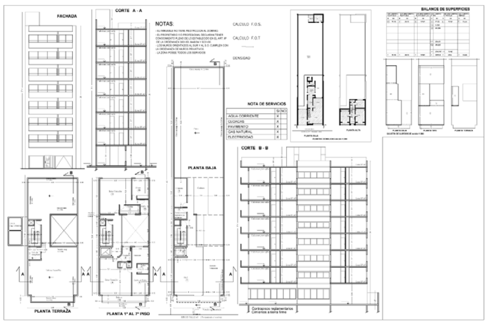
[a] Optional

user must set it. The column is instantiated as *IfcColumn* class of IFC. Additional information as column tags can also be linked, after merging OCR results.

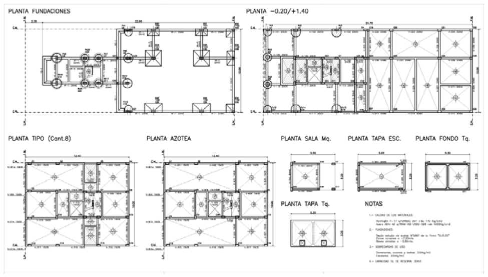
- **Beams.** This element is also restricted to concrete rectangular sections. On floor blueprints, the length and width of the element can be extracted. The height is normally indicated on the plan flow next to the element tag, which can be obtained via OCR. The beam description, which has normally "V-000 - WIDTH x HEIGHT" format on our database, must be correctly parsed. Beams are instantiated as *IfcBeam* class. Beams are supported by columns or concrete walls, but this structural relationship is not set in the model.
- **Slabs.** Concrete slab's width and length are obtained from floor plans, but its thickness has to be sourced by an OCR technique. The storey level has a unified height, and despite our dataset having more than one case with slabs on bathrooms under level, they were instanced on the same plane. Slabs are instanced as *IfcSlabs*. Slabs are supported by beams, concrete walls, columns, or concrete walls, but this structural behavior is not set on the BIM model.

Architectural plans. In Fig. 4(a) we show an example of a full architectural layout plan. The architectural plan contains a variety of elements and information, and therefore the detection of objects and semantic information extraction is more complex compared to the structural plans. We focused on walls, doors, and windows as the main elements. The doors and windows require a host, like a void element on walls, for modeling. Sanitary artifacts shall be considered in future works when mapping the plumbing system. Sometimes, architectural floors provide column locations, but structural plans override them avoiding inconsistencies.

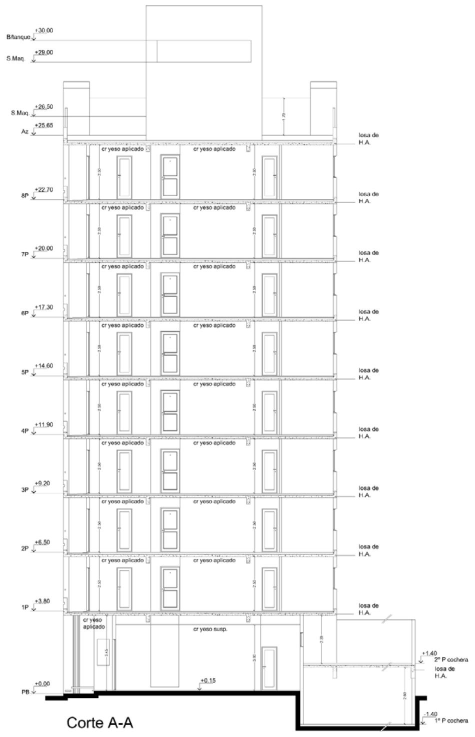
- **Walls**. This element can be represented with a variety of line styles and hatches. It is recognized as a polygon, which allows the modeling of shapes and paths variety. The width and length, or its boundary can be recognized from floor plans, but its height has to be sourced from the sections view or defined by the user in case it is not available. Walls are instantiated as *IfcWallStandardCase* class. Wall materials are not detailed in our architectural floor plan dataset.
- **Windows** and **Doors**. These elements are recognized as a rectangular box on the wall, where the width and length are obtained from floor plans. With OCR methods it is possible to extract semantic information, such as the element type or classification, considering that a carpentry detail sheet contains further design details (i.e. height and material) but the proposed method is restricted to the floor plan. Windows are instanced as *IfcWindows* but require modeling a wall and void element on the host. That is to say, the wall polygon is subtracted to get the wall opening. Windows offset from the floor and height has to be estimated. For the door case, the elements are instanced as *IfcDoors* with a simplified model without opening side considerations.



(a) Architectural blueprint layout



(b) Structural blueprint layout



(c) Section view A-A

Fig. 4. Architectural, structural, and section view blueprints on dataset.

3.3. Network architecture

We used Mask R-CNN [38] algorithm for object detection and segmentation based Mask R-CNN implementation developed by Matterport [45]. The project is open source and released under a permissive license (i.e. MIT License). The code was implemented on Python 3, Keras, and TensorFlow. The model generates bounding boxes and segmentation masks for each instance of an object

Table 2
Detail of elements labeled in the structure blueprints data set.

Structure dataset qty elements labeled				
Set	Column	Beam	Slab	Sum
Train orig	201	270	91	**562**
Train aug	622	887	262	**1771**
Test orig	105	164	42	**311**
Test aug	191	300	102	**593**
Sum	**1119**	**1621**	**497**	**3237**

in the image. It is based on Feature Pyramid Network (FPN) and a ResNet101 backbone. We used transfer learning from pre-trained weights for MS COCO[5] The same network was adopted for performing three main tasks: predicting structural elements with three labels (e.g. column, beam, and slabs), predicting the architectural elements with three labels (e.g. wall, window, and door), and a third one to predict levels height from sections view with one label for each storey. For each plan type (see Fig. 4 for examples), we trained a neural network.

3.4. Dataset setup

3.4.1. Labeling

This is a manual task performed by an engineer who is able to understand drawing elements' semantics. The engineer used Wkentaro's Labelme [46] tool, a software that allows us to label key elements on the input image and gets, as output, annotations ready to use with our selected models. In Fig. 5 a column 5(a), a beam 5(b), and 5(c) slab are tagged on the structural plan. This labeling process was performed for Architectural and Section plans as well. The labeling strategy aims to identify the elements and their shapes. In [26], the labeling covers partially the target element and includes the reference text 5(d).

On the other side, [31,32,47] labeled elements by following the element's perimeter. This leads to an element identification extracting also the morphological information with high precision like location, length, and width, or even a mask for non-rectangular shapes.

Other authors [26,28] included textual annotations along with the drawing elements and relied on the drawing grid to infer the element length. This means that the approach excludes plans without any grid from the scope. Moreover, the method does not allow extracting morphological information for non-rectangular shapes.

3.4.2. Augmentation

The augmentation technique allows the creation of new images from existing ones by performing transformations to the original image. It is possible, for example, to rotate, add zoom, change colors or lighting, add noise, etc. This technique is really useful when we have few images or when they are difficult to label. Its importance lies in the fact that it allows for increasing the effectiveness of the model since a large number of extra images contemplate different conditions or situations that were indeed not contemplated in the original images. will be provided and it is also useful to avoid *overfitting*, that is, the memorization by the model of the training images, which leads to low accuracy in detecting new images.

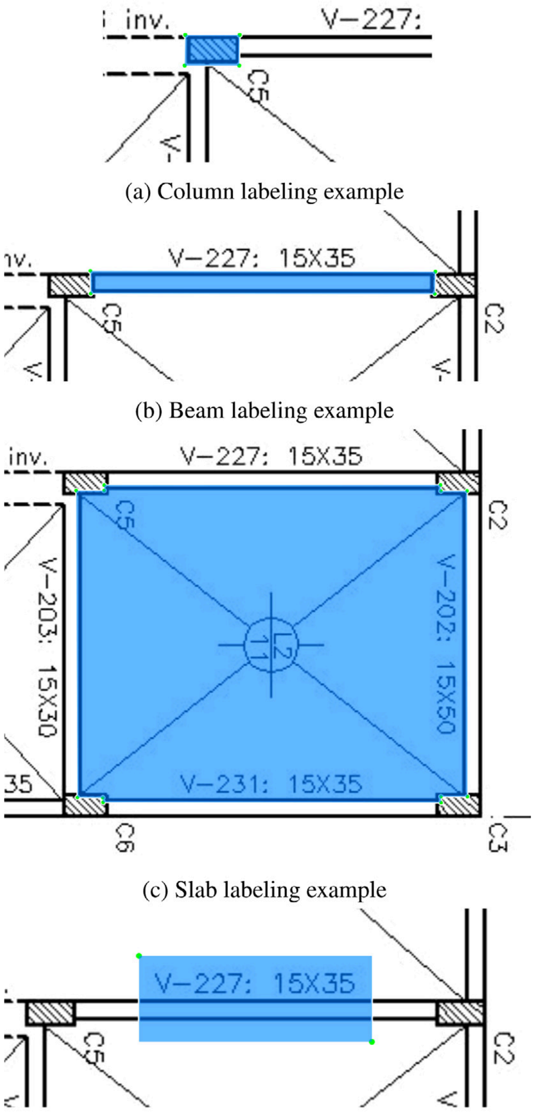
Modifying an image means that we also have to update its mapping file since the same transformation is applied to the label. Our augmentation process involves taking the dataset and for each pair of (image, label) converting the label file to a .csv file for easier management of bounding boxes and using a Python library called imgaug.[6] The outcome of the augmentation approach is a larger set of images that are later used for training and testing purposes. The performed transformations were rotation, scale (up and down), crop, flip (horizontal and vertical), noise, and blur.

3.5. Dataset separation

After labeling the images, they must be divided into 2 folders (training and test). The first will be used for training, and the second to periodically evaluate training performance, allowing the algorithm to make the necessary adjustments to the network to improve predictions. The percentage of distribution between both folders was 75% for training and 25% for the test. For BLD-AR, 69 images were used for training, with 17 originals and 52 augmented. On the other hand, 22 images were assigned for testing, composed of 8 originals and 14 augmented. The BLD-ST dataset was split in the same way. As presented in Table 2, within training and test images, the total elements labeled are 873, and including augmented images is 3237 elements: 1119 columns, 1621 beams, and 497 slabs. In the same way, in Table 3, the total elements labeled are 867 where 3337 elements are obtained after applying the augmentation process: 609 windows, 705 doors, and 2013 walls.

within training and test images,

[5] COCO Dataset - https://github.com/cocodataset/cocoapi.git
[6] Imgaug - https://imgaug.readthedocs.io/en/latest/



(a) Column labeling example

(b) Beam labeling example

(c) Slab labeling example

(d) Beam labeling not adopted

Fig. 5. Structural elements labeling.

Table 3
Detail of elements labeled in the architectural blueprints data set.

Architectural dataset qty elements labeled				
Set	Window	Door	Wall	Sum
Train orig	129	139	428	**696**
Train aug	399	456	1232	**2087**
Test orig	29	39	103	**171**
Test aug	52	71	250	**373**
Sum	**609**	**705**	**2013**	**3327**

3.6. Network configuration

In this article, our main interest lies in the methodology for generating a BIM model. We have opted for the Mask R-CNN model as described in Section 3.3, but we will not focus on an exhaustive analysis of the performance of the network, to which in principle we have not made any contributions in this publication.

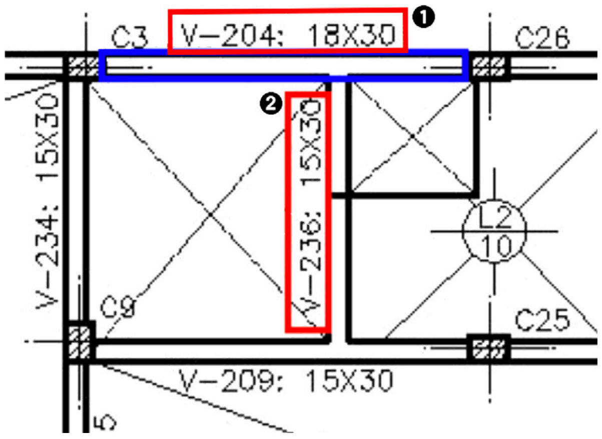


Fig. 6. Annotation processing.

3.7. Semantic information extraction

3.7.1. OCR recognition

Our approach requires the extraction of object annotations using an OCR to enrich their semantic information. Although there are world-class OCR services, we chose Google's Cloud Vision service to extract all the text presented in the image, because it has a good performance and was easily integrated into our solution by means of API requests. In [48], the OCR portion of the process was carried out by detecting horizontal and vertical text separately and generating spreadsheet files with the text data (text without image position coordinates). We instead opted to process the image and extract every text and its coordinates found on the input image. For such a task, we designed an algorithm able to match the detected blueprint elements with their corresponding label extracted via OCR.

This way we obtain accurate labels to then enrich our model with the necessary context.

3.7.2. Object's description association

The script expects two lists of elements, in this case, Labels and DetectedObjects. The script will iterate over the list of pivot elements (i.e. DetectedObjects) and search for the closest label in the list of available labels. This is done by calculating the minimum distance between the nearest points of each element. Once we have the nearest one detected, we bind them create a pair and eliminate the label from the available list. This process is repeated until there are no more matches to be made.

In Fig. 6, let us consider the marked blue column as a pivot. It would iterate over the labels (❶ and ❷) and calculate the nearest distance between them, it will then pair up the column with label ❶ since they are closest to each other.

To minimize false positives, the algorithm just does not compute the distance among all elements and all text labels, instead, it filters each label to only match the ones that correspond to the object type. This is possible due to each element type having a standard nomenclature during labeling, i.e. beams are V-XXX and columns are C-XXX. This step allowed us to reduce the number of missed assignments to a more than acceptable-value.

3.7.3. Elements associations

When detecting objects in plans, there are elements that require establishing semantic relationships with other elements. For example, when a beam is supported by columns that transmit its loads. This relation has to be inferred from the object disposition in the plan. To address this situation, a script was developed that proposes elements' relationships based on their placements. Conceptually, the approach does not differ much from the one described in Section 3.7.2, where the descriptive text of the element is associated with the element.

For our use case, we needed to know what beams are related to a certain column. Nonetheless, the module it is flexible and was built with the option to be extended.

Since each element is already classified, by simply indicating what type of element the script should use as the pivot, the script can search and assemble the relation between elements. The script considers objects to be related by calculating the distance between the nearest points each other. If it is within a threshold, 20 pixels to be precise, the objects will be considered related and that relationship will be recorded. This process is repeated for each object of the type selected as the pivot. Additionally, we considered a column to have a maximum of 4 beams assigned, and beams to have a maximum of 2 columns. This is done in order to prevent over-assignments and minimize the impact of missing/duplicate detection that might possibly occur during the object detection.

3.8. Building information model generation

As mentioned, the result obtained after identifying architectural and structural objects on blueprints, are TXT files with the detected elements as a collection dictionary with keys-values pair for defining the element class and the boundary box pair points defined as X,Y pixels. The objects to be recognized require volume definition in order to create them on an IFC model.

The sets of files are the inputs for the BIM generation model algorithm developed in Python. Each type of input is kept in a specific folder. The input files must be organized in such a way that there is a uniqueness between the position of each storey in its respective folder, even if one of them does not contain data. hlThe first step consists of parsing the detected structure elements stored in a file, where each file represents a storey. The storey height is set based on a parameter in our script. But the approach could be enhanced to process section view plans and pull the storey height from it. For each storey level, the set of each element type as columns, beams, and slabs are instantiated.

Given that blueprints are two dimensional documents, we defined two possible strategies to set the columns' heights. If a document scale is detected (pixels to meters), we use a constant value for the column's height (i.e. 2.60 m). This can be improved by processing x-sections plans to get the storey's levels. If the scale is not detected, we used a ratio coefficient as height = 13.2 times the minor dimension of all detected columns. We have set this relationship considering that the local regulation defines 20 cm as a minimum dimension for reinforced concrete columns; it is 2.64 meters in this case. The architectural prediction input is merged in a similar way, including in the array of elements such as walls, doors, and windows. The next step is to run the BIM Generation script with the input array as a result of merging input files, which creates new instances of each column, beam, slab, wall, door, and window, and generates one IFC model file. For manipulating the IFC file, we used an IfcOpenShell open-source (LGPL) software library for python. At the moment, input files only contain geometry data. The model can be enriched by identifying other properties like material by combining the approach where the material is extracted from indoor/outdoor building pictures [49]. The IFC BIM structure model is created by using *IfcColumn*, *IfcBeam*,*IfcSlab*. Architectural model components were created by using *IfcWallStandardCase*, *IfcDoor*,*IfcWindow*. The Doors and windows belong to a family object that requires a host element. In this case, for each instance of door or windows, the script creates a wall and opening, as *IfcOpeningElement*, where the carpentry is related. A standard template for an *IfcProject* is used for setting up quickly project information, units system, and directions. New entity instances will be related to the *IfcProject*. IFC model implements the composition/decomposition to represent the relationship among the building elements. The aggregation relationship *IfcRelAggregates*[7] is a special type of the general composition/decomposition (or whole/part) relationship.

Fig. 7 illustrates the steps to create the object instance for IfcColumnStandardCase[8] that includes the OCR description association stage. After recognizing the object with the trained neural network ❶, the text extracted via OCR❷ is associated by proximity to the element ❸, as described in Section 3.7.2. The step ❹ consists of the application of the alignment and scale transformations, which allows mapping from the Pixel Coordinate System (PCS) to the Drawing Coordinate System (DCS). The plant and the height of the element are considered in sequence when the IFC object is instantiated ❺. Continuing on ❻, the element profile is defined by a sequence of points specified as Python tuples retrieved from the *input TXT file* as a polyline *IfcPolyline* which defines a closed profile *IfcArbitraryClosedProfileDef* or swept area. The solid volumes are defined as extruded area solid (*IfcExtudedAreaSolid*) by sweeping a bounded planar surface through a given direction (*ExtrudedDirection*) and the length of the extrusion given by a Depth attribute. The position of the extruded area solid is defined by its location (IfcAxist2Placement3D object), for the columns the extrusion direction has been defined as a vertical vector (IfcDirecton), and the extrusion depth is defined by the column height. The position of the extruded area solid is defined by its location (IfcAxist2Placement3D object), the extrusion direction as a vertical vector (IfcDirecton object), and the extrusion depth by the column height.

A body representation is related to the swept solid (*IfcShapeRepresentation*) to get a product shape (*IfcProductDefinitionShape*). The product to be instanced as, for example, *IfcColumn* ❽, is associated with the product shape, material ❼ (*IfcMaterial*), property set (*IfcPropertySet*), and element quantities (*IfcElementQuantity*) once the area is extruded (*IfcProductDefinitionShape*). For the sake of simplicity, the process of modeling the beam element indicated in ❾ is similar to that described for the column case explained in steps 1–8. The column and the beam are related (*IfcRelDedineseByType*) to each other in step ❿.

In the case of window or door, additionally has to be instance a *IfcOpeningElement* and relate it to a wall host before relating the product to the opening.

4. Case study: From blueprints to BIM

4.1. Dataset

To train our network, we prepared two datasets, namely BLD-AR and BLD-ST for building architecture blueprints and building structure blueprints respectively. Both datasets include real-life building projects with less than 10 levels provided by the building constructor. Figure Fig. 8 presents a detail of the composition of the dataset: 9 concrete buildings with 72 structural (BLD-ST) and 38 floorplan images, 18 sections views, and 9 front/back views (BLD-AR). Floor plans are mostly rectangular shapes. The set BLD-AR consists of building architecture like floorplans, building sections, front view, and back view, gathered in the same plan layout (Fig. 4(a)), which was split into floor plans (Fig. 2), and views (Fig. 4(c)) before tagging. The set BLD-ST consists of floor plans with structural elements such as columns, beams, slabs, and foundations, and are required to be separated per floor plan.(Fig. 2(c))

[7] https://standards.buildingsmart.org/IFC/DEV/IFC4_3/RC1/HTML/schema/ifckernel/lexical/ifcrelaggregates.htm

[8] https://standards.buildingsmart.org/IFC/DEV/IFC4_3/RC1/HTML/schema/ifcsharedbldgelements/lexical/ifccolumnstandardcase.htm

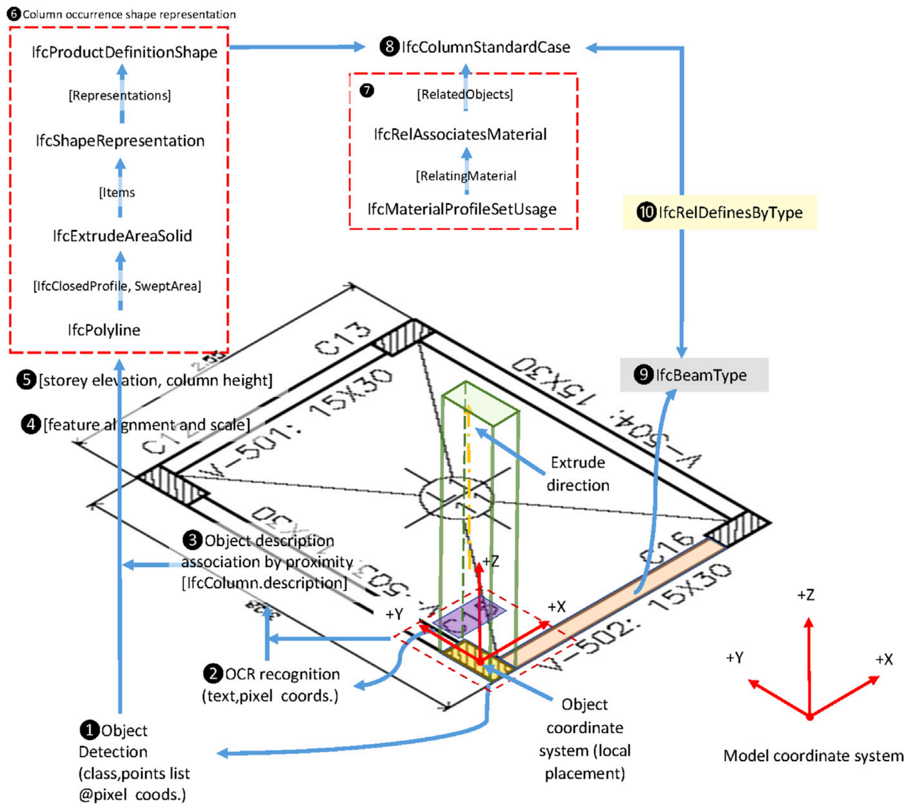
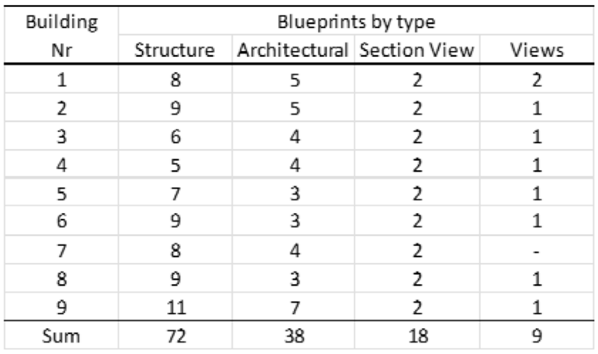

6 Column occurrence shape representation
IfcProductDefinitionShape
[Representations]
IfcShapeRepresentation
[Items
IfcExtrudeAreaSolid
[IfcClosedProfile, SweptArea]
IfcPolyline
8 IfcColumnStandardCase
7 [RelatedObjects]
IfcRelAssociatesMaterial
[RelatingMaterial
IfcMaterialProfileSetUsage
10 IfcRelDefinesByType
9 IfcBeamType
5 [storey elevation, column height]
4 [feature alignment and scale]
3 Object description association by proximity [IfcColumn.description]
2 OCR recognition (text,pixel coords.)
1 Object Detection (class,points list @pixel coods.)
Extrude direction
Object coordinate system (local placement)
Model coordinate system
+Z
+Y
+X
C13
C12
C16
V-501: 15X30
V-502: 15X30
V-504: 15X30

Fig. 7. IFC Column Standard Case implementation flow.



Building Nr	Blueprints by type			
	Structure	Architectural	Section View	Views
1	8	5	2	2
2	9	5	2	1
3	6	4	2	1
4	5	4	2	1
5	7	3	2	1
6	9	3	2	1
7	8	4	2	-
8	9	3	2	1
9	11	7	2	1
Sum	72	38	18	9

Fig. 8. Detail of blueprints data set.

4.2. Training model configuration

We used Google Colab Pro service to train all models involved in our approach. In order to analyze the performance of the models, we tested different hyperparameters settings comprising the *Number of Training Steps per Epoch*, *Number Epoch*, and *Batch Size* parameters. We computed the mean Average Precision (mAP) for each setting and the time it took to train the model. The sessions shared some parameters like the learning rate (0.0001), and validation steps (10); the images were picked randomly in each step.

The Table 4 shows the batch size, training steps, validation steps, number of samples ($batch * epochs * steps$), the time it took the training, and mAP for the six training sessions. In the first scenario (batch size 1, 20 steps, 2500 epochs), the mAP was 96.77% and required 7 h of training. From Session 2 to Session 6, we increased the training step from 20 to 50, batch size from 1 to 6, and resumed the number of epochs from the last saved checkpoint in every new training session. In this way, we were able to detect where the mAP gets stable on session 6 where it was 96.32%.

Table 4
Training sessions.

#	GPU	Batch	Training steps	Validation steps	Epoch	# samples	Time – h	mAP
1	NVIDIA A100	1	20	10	2500	50000	7	0.967
2	NVIDIA T4	6	50	10	11	3300	12	0.876
3	NVIDIA T4	6	50	10	17	5100	24	0.927
4	NVIDIA T4	6	50	10	24	7200	36	0.944
5	NVIDIA T4	6	50	10	34	10200	48	0.958
6	NVIDIA T4	6	50	10	38	11400	60	0.963

Although it is recommended to set *# of images/ batch size* as training steps to train all over the dataset samples, we were not able to run such a number of steps on Google Colab Pro and our sessions were closed without even one checkpoint. That is to say, each epoch took more than 12 h. Consequently, we reduced the number of steps per epoch to 50 which allowed completing the epoch safely. The reader can notice that there is a huge difference between Session 1 and the others because, unfortunately, the allocated GPU (with high computational power) was not available in subsequent sessions; this significantly increased the number of training hours required for similar training.

The latest session required processing less number of images ($\#samples$ column) than the first scenario to get a similar mAP.

Without a doubt, a topic for future work is to compare the benchmarks obtained, with those resulting from the use of other CNN models, without ruling out different configurations that would improve the fine-tuning of the Mask R-CNN model used in this research.

4.3. Cloud network training

The network was trained using the Google Colab Pro platform; although resources are not guaranteed, we were assigned NVIDIA T4 and A100[9] with priority access and high-RAM setting at this time. In general, notebooks can run for at most 12 h, depending on availability and usage patterns. For processing the classified element, we developed custom scripts based on Python 3.8.16 programming language. The original resolution of the images for the input floor plan was around 3000 x 7000. We resized the images before training because that image size was not processable by the model without overflowing the available resources.

4.4. Running example

The building to be modeled has 9 levels for functional units, and 4 additional levels that correspond to the machine room and elevated tank. Level 0 (ground floor) corresponds to the building access, levels 1 to 8 correspond to repeated standard floor plans and each floor of the building matches a rectangular shape, resulting in a quite simple volume with repeated levels. The output files obtained from both models trained previously were 13 for structure and 12 for architecture. We ran our Code described in Section 3.8 using the IfcOpenShell python library for instantiating elements. The script processes in alphanumeric order each structure file and then the architectural ones. The alphanumerical ordering of the files coincided with the location of each floor from the lowest to the highest level.

The model result is an IFC file that can be presented visualized with xBIM Xplorer,[10] a free open-source IFC viewer. A screenshot of the complete model is presented in Fig. 9, already integrated with the architectural and structural elements. In Fig. 10 an x-view of the same building is shown; the inner elements modeled as doors, windows, columns, and walls can be appreciated. Fig. 11 presents the result of running our approach using only structural plans where columns, beams, and slabs were instantiated. It is highlighted that a key factor to modeling a building coherently turns out to be the consistency between vertical positions and their appropriate scale between different levels of the element detected by each trained model that maps the structure plane as architecture respectively. This key factor is considered in Section 3.1 feature alignment and scale. To generate the model, 1155 mapped elements were instantiated: columns (181), slabs (110), beams (321), walls (391), doors (103), and windows (49). It is worth mentioning that minor adjustments were made on some elements to get a clear model. For example, slab stretching connects all the beams.

5. Discussion

In this section, we discuss the benefits, challenges, and limitations of our approach.

[9] https://cloud.google.com/compute/docs/gpus
[10] https://docs.xbim.net/downloads/xbimxplorer.html

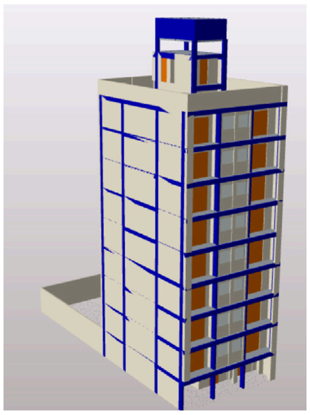


Fig. 9. Model Generated — 3D view.

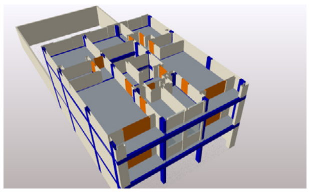


Fig. 10. Floor x-view.

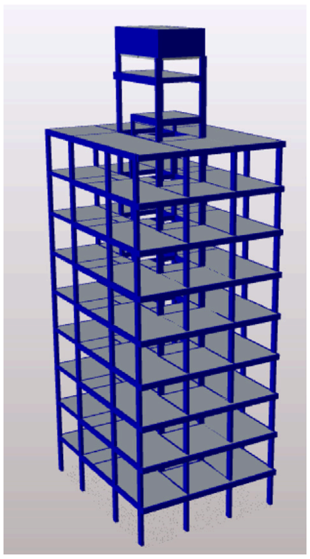


Fig. 11. Structure modeled.

5.1. Pros

Our approach to generating BIM from blueprints aims at generating full BIM models from plans without any manual tasks. The digitalization of large datasets of building plans would be faster (milliseconds) and cheaper than a manual translation of existing designs by a specialist (minutes to hours). In the context of a city hall that has decades of plans and building permits, bulk processing is feasible but it is not the case for the manual translation to BIM. Capturing the experience and knowledge available in plans into models allows performing big data analysis to train a neural network that could aid the design of new buildings based on recommendations.

5.2. Cons

In our approach, once the elements are identified, there is a mapping script that generates the right IFC element. We have only covered a basic set of IFC elements to test our approach. A building design is an art that continuously explores different element combinations to get the most functional but creative solution. Our approach has not been yet tested on different building types and locale-specific design styles.

5.3. Limitation

There are some limitations of the current solution. First, this work showed the benefits of applying ML to generate preliminary BIM models that help address some analysis concerns. There are works that address the code of practice compliance checking [50] in BIM based on local requirements. The generated models for existing buildings may fail the validation process if they were developed using a previous code of practice that is incompatible.

Second, the lack of a larger blueprint dataset restricts the ability to process diverse building types. Our dataset consists only of concrete buildings. The extended dataset should include, but not be limited to, houses, industries, and warehouses built of concrete, steel, wood, and/or other material. For example, we have evaluated the proposed method considering a basic subset of BIM elements however it is still pending to pull fine-grain details to increase the Level of detail (LoD) [51]. The level of detail is LOD 200 (walls, doors, and windows) and LOD 290 [52] (slab, column, and beams), where the model element is graphically represented as a generic system or object, with approximate quantities, size, shape, and their exact positions subject to change (i.e. windows height). Non-graphical information may also be included in the modeled elements. From the generated model, its element's properties can be checked out such as geometric dimensions, area, and volume audited. Moreover, the structural model could be enhanced by integrating this approach with [53] where appendix tables that include reinforcement details information. These tables would be ingested and combined with beams and slab elements, thereby increasing the LOD and allowing for sophisticated analysis, such as building earthquake studies. One of the main goals of this approach is to process the plans in bulk, in order to learn the general aspects of buildings. The generated model still lacks many details such as materials, HVAC systems, plumbing systems, etc., although the approach can be extended to consider those details. As discussed before, their extraction depends on the local drawing/design style.

There is a natural limitation that requires building plans' availability as digital assets. Municipal archives must implement digitization techniques within their internal processes to generate these assets. Customizing the generation script of each new type of element or class to be modeled is a task that requires qualified personnel.

Training a neural network for supporting all possible design elements and adjusting the script for combining identified elements is complex. So we suggest prioritizing the most important elements to get a simple but mindful BIM model. For example, modeling the layer composition of a wall is more relevant than modeling the internal furnishings if you want to perform a thermal balance analysis. The modeled building has a simple and repetitive geometry in height. Floors with curved or sloped beams and sloped ceilings have not been tested. Within existing constructions, some have been through a continuous reform process and may have properly documented the building history. Therefore, plans from different periods can be processed, but this is beyond the scope of this work since they imply modeling the different stages for the same construction, from a previous model. It is important to clarify that the generated model tries to reflect the processed documentation as it is. In this work, we did no consider any Code of Practice for e-summit applicable to the generated BIM model.

6. Conclusions and further work

The generation of building models from plans has been widely studied. Recently, with the advancement of deep learning models, different works have been presented that seek to attack the same problem from the machine learning perspective. The proposed methodology uses different sources to generate an object-oriented model at a low cost and supports the BIM model generation in an IFC format from floor plan blueprint images using machine learning. The approach allows the plans to be processed in bulk for obtaining BIM with room for increasing the LoD. The methodology is novel as it introduces instance segmentation by implementing Mask R-CNN model, trained on a novel dataset BLD-AR and BLD-ST. The final result is an IFC interchangeable file generated with open-source libraries.

In this work, we presented an approach to generate a multi-level building model where beams, columns, slabs, windows, walls, and doors are instantiated, in their standard cases. We could show the preliminary results of an object recognition algorithm to process structural and architectural blueprints automatically.

The generation of BIM models from existing construction documentation undoubtedly will be targeted as an important subject by data science. Some of the benefits of this approach are the possibility to:

- Aiding retrofit design tasks for an existing building with recommendations based on BIM models and massive building analysis to optimize energy consumption is of great interest.
- Aiding design tasks with recommendations based on BIM models and massive building analysis to optimize energy consumption is of great interest.

We plan to gather blueprints honoring different drawing styles and from different countries applying different standards. We are also planning to synthesize blueprints for the training step from existing BIM models [41] using different drawing styles. As this work was limited to structural and architectural drawings, we are also planning to study how to identify mechanical, electrical, and plumbing elements. We plan to research how to automatically automate the classification of plans and their content in bulk mode. In order to evaluate the model quality of this solution. we plan to request architects/engineers to design BIM models and compare the manual and system-generated versions to find possible optimizations and missing best practices. Additionally, due to the computational power required to train the neural network, we will work on a new approach based on a hybrid solution combining small and big element detection [54]. The approach splits the image into tiles and uses a specialized neural network trained with small elements such as doors or columns and bigger ones as slabs or long walls.

CRediT authorship contribution statement

Martin Urbieta: Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing. **Matias Urbieta:** Conceptualization, Methodology, Supervision, Writing – original draft, Writing – review & editing. **Tomas Laborde:** Software, Writing – original draft. **Guillermo Villarreal:** Software, Writing – original draft. **Gustavo Rossi:** Supervision, Writing – review & editing.

Declaration of competing interest

There is no conflict of interest.

Data availability

Data will be made available on request

References

[1] ISO Standard, Building Information Modeling—Information Delivery Manual—Part 1: Methodology and Format, ISO 29481-1: 2016 (E), International Organization for Standardization, Geneva, CH, 2016.
[2] ISO Standard, Industry Foundation Classes (IFC) For Data Sharing in the Construction and Facility Management Industries — Part 1: Data schema, ISO 16739-1:2018, International Organization for Standardization, Geneva, CH, 2018.
[3] Kathryn E. Cassino, Harvey M. Bernstein, F. Asce, Leed Ap, Michele A. Russo, Ap Editorial Advisor, Chief Author, Stephen A. Jones, Donna Laquidara-Carr, William Taylor, Juan Ramos, Art Director, Alison Lorenz, Terumasa Yamada, Bruce Buckley, Deborah Snoonian Glenn, Katharine Logan, Jeffrey Yoders, Design and construction intelligence SmartMarket report McGraw hill construction the business value of BIM for owners SmartMarket report executive editor, 2014.
[4] Brittany K. Giel, Raja R.A. Issa, Return on investment analysis of using building information modeling in construction, J. Comput. Civ. Eng. 27 (5) (2013) 511–521.
[5] P. Gerbert, S. Castagnino, C. Rothballer, A. Renz, R. Filitz, Digital in engineering and construction, 2016, pp. 1–22, The Boston Consulting Group.
[6] Letizia D'Angelo, Magdalena Hajdukiewicz, Federico Seri, Marcus M. Keane, A novel BIM-based process workflow for building retrofit, J. Build. Eng. 50 (2022) 104163.
[7] Maggie Khaddaj, Issam Srour, Using BIM to retrofit existing buildings, Procedia Eng. 145 (2016) 1526–1533, ICSDEC 2016 – Integrating Data Science, Construction and Sustainability.
[8] Francesca Noardo, Dogus Guler, Judith Fauth, Giada Malacarne, Silvia Mastrolembo Ventura, Miguel Azenha, Per-Ola Olsson, Lennart Senger, Unveiling the actual progress of digital building permit: Getting awareness through a critical state of the art review, Build. Environ. 213 (2022) 108854.
[9] Kaleem Ullah, Emlyn Witt, Irene Lill, The BIM-based building permit process: Factors affecting adoption, Buildings 12 (1) (2022).
[10] Qiuchen Lu, Ajith Kumar Parlikad, Philip Woodall, Gishan Don Ranasinghe, Xiang Xie, Zhenglin Liang, Eirini Konstantinou, James Heaton, Jennifer Schooling, Developing a digital twin at building and city levels: Case study of west cambridge campus, J. Manage. Eng. 36 (3) (2020) 05020004.
[11] R. Kippers, Mila Koeva, M. Keulen, Sander Oude Elberink, Automatic 3d building model generation using deep learning methods based on cityjson and 2D floor plans, Int. Arch. Photogram. Remote Sens. Spatial Inf. Sci. XLVI-4/W4-2021 (2021) 49–54.
[12] Ravi Peters, Balázs Dukai, Stelios Vitalis, Jordi van Liempt, Jantien Stoter, Automated 3D reconstruction of LoD2 and LoD1 models for all 10 million buildings of the netherlands, Photogramm. Eng. Remote Sens. 88 (2022) 165–170.
[13] Luís Sanhudo, Nuno M.M. Ramos, João Poças Martins, Ricardo M.S.F. Almeida, Eva Barreira, M. Lurdes Simões, Vítor Cardoso, Building information modeling for energy retrofitting – A review, Renew. Sustain. Energy Rev. 89 (2018) 249–260.
[14] Ngoc-Son Truong, Duc Long Luong, Quang Trung Nguyen, BIM to BEM transition for optimizing envelope design selection to enhance building energy efficiency and cost-effectiveness, Energies 16 (10) (2023).
[15] Tianzhen Hong, Zhe Wang, Xuan Luo, Wanni Zhang, State-of-the-art on research and applications of machine learning in the building life cycle, Energy Build. 212 (2020) 109831.
[16] Cheng Zhang, Yang Zou, Johannes Dimyadi, A systematic review of automated BIM modelling for existing buildings from 2D documentation, in: ISARC. Proceedings of the International Symposium on Automation and Robotics in Construction, Vol. 38, IAARC Publications, 2021, pp. 220–226.
[17] Pablo N. Pizarro, Nancy Hitschfeld, Ivan Sipiran, Jose M. Saavedra, Automatic floor plan analysis and recognition, Autom. Constr. 140 (2022) 104348.
[18] Lucile Gimenez, Sylvain Robert, Frédéric Suard, Khaldoun Zreik, Automatic reconstruction of 3D building models from scanned 2D floor plans, Autom. Constr. 63 (2016) 48–56.
[19] Junfang Zhu, Hui Zhang, Yamei Wen, A new reconstruction method for 3D buildings from 2D vector floor plan, Comput.-Aided Des. Appl. 11 (2014) 704–714.
[20] Chen Liu, Jiajun Wu, Pushmeet Kohli, Yasutaka Furukawa, Raster-to-vector: Revisiting floorplan transformation, in: 2017 IEEE International Conference on Computer Vision, ICCV, 2017, pp. 2214–2222.
[21] Bin Yang, Boda Liu, Dayu Zhu, Binghan Zhang, Zhichen Wang, Ke Lei, Semiautomatic structural BIM-model generation methodology using CAD construction drawings, J. Comput. Civ. Eng. 34 (3) (2020) 04020006.

[22] Brandon Bortoluzzi, Ivan Efremov, Clarice Medina, Daniel Sobieraj, J.J. McArthur, Automating the creation of building information models for existing buildings, Autom. Constr. 105 (2019) 102838.
[23] Mengtian Yin, Llewellyn Tang, Tongyu Zhou, Ya Wen, Ruohan Xu, Wu Deng, Automatic layer classification method-based elevation recognition in architectural drawings for reconstruction of 3D BIM models, Autom. Constr. 113 (2020) 103082.
[24] Qiuchen Lu, Sanghoon Lee, A semi-automatic approach to detect structural components from cad drawings for constructing as-is bim objects, in: Congress on Computing in Civil Engineering, Proceedings, 2017, pp. 84–91.
[25] Qiao Wen, Rui-Guang Zhu, Automatic generation of 3D building models based on line segment vectorization, Math. Probl. Eng. 2020 (2020) 1–16.
[26] Yunfan Zhao, Xueyuan Deng, Huahui Lai, A deep learning-based method to detect components from scanned structural drawings for reconstructing 3D models, Appl. Sci. (Switzerland) 10 (2020).
[27] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: Unified, real-time object detection, in: Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 2016-December, 2016, pp. 779–788.
[28] Yunfan Zhao, Xueyuan Deng, Huahui Lai, Reconstructing BIM from 2D structural drawings for existing buildings, Autom. Constr. 128 (2021) 103750.
[29] R. Girshick, Fast R-CNN, in: Proceedings of the IEEE International Conference on Computer Vision, Vol. 2015 International Conference on Computer Vision, ICCV 2015, 2015, pp. 1440–1448.
[30] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, Adv. Neural Inf. Process. Systems 28 (2015) 91–99.
[31] Zhiliang Zeng, Xianzhi Li, Ying Kin Yu, Chi-Wing Fu, Deep floor plan recognition using a multi-task network with room-boundary-guided attention, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, arXiv, 2019, pp. 9096–9104.
[32] Jihyo Seo, Hyejin Park, Seungyeon Choo, Inference of drawing elements and space usage on architectural drawings using semantic segmentation, Appl. Sci. 10 (20) (2020).
[33] Samuel Dodge, Jiu Xu, Björn Stenger, Parsing floor plan images, in: 2017 Fifteenth IAPR International Conference on Machine Vision Applications, MVA, IEEE, 2017, pp. 358–361.
[34] Chenxi Liu, Alexander G. Schwing, Kaustav Kundu, Raquel Urtasun, Sanja Fidler, Rent3D: Floor-plan priors for monocular layout estimation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 3413–3421.
[35] Ahti Kalervo, Juha Ylioinas, Markus Häikiö, Antti Karhu, Juho Kannala, Cubicasa5k: A dataset and an improved multi-task model for floorplan image analysis, in: Scandinavian Conference on Image Analysis, Springer, 2019, pp. 28–40.
[36] Lluís-Pere de las Heras, Oriol Ramos Terrades, Sergi Robles, Gemma Sánchez, CVC-FP and SGT: A new database for structural floor plan analysis and its groundtruthing tool, Int. J. Document Anal. Recognit. (IJDAR) 18 (1) (2015) 15–30.
[37] Yijie Wu, Jianga Shang, Pan Chen, Sisi Zlatanova, Xuke Hu, Zhiyong Zhou, Indoor mapping and modeling by parsing floor plan images, Int. J. Geogr. Inf. Sci. 35 (6) (2021) 1205–1231.
[38] Kaiming He, Georgia Gkioxari, Piotr Dollár, Ross Girshick, Mask r-cnn, in: Proceedings of the IEEE International Conference on Computer Vision, arXiv, 2017, pp. 2961–2969.
[39] Jonathan Long, Evan Shelhamer, Trevor Darrell, Fully convolutional networks for semantic segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, arXiv, 2014, pp. 3431–3440.
[40] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, Serge Belongie, Feature pyramid networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, arXiv, 2016, pp. 2117–2125.
[41] Tanya Bloch, Rafael Sacks, Comparing machine learning and rule-based inferencing for semantic enrichment of BIM models, Autom. Constr. 91 (2018) 256–272.
[42] Stefano Zorzi, Ksenia Bittner, Friedrich Fraundorfer, Machine-learned regularization and polygonization of building segmentation masks, in: 2020 25th International Conference on Pattern Recognition, ICPR, 2021, pp. 3098–3105.
[43] Richard Szeliski, Feature-based alignment, in: Computer Vision: Algorithms and Applications, Springer London, London, 2011, pp. 273–301.
[44] Will Y. Lin, Automatic generation of high-accuracy stair paths for straight, spiral, and winder stairs using IFC-based models, ISPRS Int. J. Geo-Inf. 9 (4) (2020).
[45] Waleed Abdulla, Mask R-CNN for object detection and instance segmentation on keras and TensorFlow, 2017, GitHub repository, Github, https://github.com/matterport/Mask_RCNN.
[46] Kentaro Wada, mpitid, Martijn Buijs, Zhang Ch. N., Bc. Martin Kubovč ík, Wkentaro/labelme: v4.6.0, 2021, Zenodo.
[47] Weixin Huang, Hao Zheng, Architectural drawings recognition and generation through machine learning, in: Recalibration on Imprecision and Infidelity - Proceedings of the 38th Annual Conference of the Association for Computer Aided Design in Architecture, ACADIA 2018, 2018, pp. 156–165.
[48] Qiuchen Lu, Long Chen, Shuai Li, Michael Pitt, Semi-automatic geometric digital twinning for existing buildings based on images and CAD drawings, Autom. Constr. 115 (2020) 103183.
[49] Qiuchen Lu, Sanghoon Lee, Long Chen, Image-driven fuzzy-based system to construct as-is IFC BIM objects, Autom. Constr. 92 (2018) 68–87.
[50] Aimi Sara Ismail, Kherun Nita Ali, Noorminshah A. Iahad, A review on BIM-based automated code compliance checking system, in: 2017 International Conference on Research and Innovation in Information Systems, ICRIIS, 2017, pp. 1–6.
[51] BIMForum, Level of Development (LOD) Specification Part I & Comentary, BIM Forum, 2020.
[52] Petteri Uusitalo, Olli Seppänen, Eelon Lappalainen, Antti Peltokorpi, Hylton Olivieri, Applying level of detail in a BIM-based project: An overall process for lean design management, Buildings 9 (5) (2019).
[53] Nandhinee P.R., Harinath Krishnamoorthy, Koushik Srivatsan, Anil Goyal, Sudarsun Santhiappan, DEXTER: An end-to-end system to extract table contents from electronic medical health documents, 2022, arXiv:2207.06823.
[54] F. Ozge Unel, Burak O. Ozkalayci, Cevahir Cigla, The power of tiling for small object detection, in: IEEE Computer Society Conference on Computer Vision and Pattern recognition Workshops, Vol. 2019-June, 2019, pp. 582–591.