Transforming 3D BIM Data into Tensor Formats for ML Applications in Architecture

Image stacking method

Arman Yasa[1], Togan Tong[2]
[1,2]Yildiz Technical University, Department of Architecture
[1]arman.yasa@std.yildiz.edu.tr [2]tong@yildiz.edu.tr

A critical factor for getting good results in machine learning (ML) applications is the presence of good quality and especially large datasets. ML is most especially applicable to the integration of the multi-dimensional aspects of architectural design. However, the current datasets are presented in various formats, which have inhibited the creation of large-scale datasets and pre-trained ML models. Furthermore, the current reliance on 2D architectural data is not enough to fulfill the data needs of architectural ML models, because design and the final outputs of architectural design are both 3D. BIM data could help to solve this issue since it contains rich geometric information as well as structural and spatial information. This paper is dedicated to the process of converting 3D BIM data for use as tensor data in ML applications. The proposed conversion method uses image stacking, which is used in medical imaging applications like MRI, CT scans and engineering applications. The paper gives a comprehensive account of how the geometric and class data from BIM is converted to image stack format at different levels of resolution, and how class data is encoded into pixels.
This method is expected to help in the integration of 3D architectural representations into the machine learning environment in order to develop big data sets and pre-trained models. Also,BIM data is more diverse than other sources, it will help in identifying different relationships within the context of architectural machine learning.

Keywords: *BIM, Big Data, Machine Learning, Image Stacking.*

INTRODUCTION

The application of machine learning (ML) tools has opened new research areas and possibilities in the field of architectural design. Since architectural design is defined with many elements, parameters and relationality in the process and product, ML tools have a wide potential (Topuz and Alp., 2023). However, the rate of usage and development of ML and AI tools developed for architectural design is quite low as compared to other fields (LLMs, Image Processing, Medical). This can be attributed to several factors such as the users of the solutions, the importance of the problems, the possibilities of digitalization, the economic return, and the popularity of the fields of study. In addition to these, another important parameter that constitutes the focus of this study is big data sets. The key factor that makes ML tools successful in achieving their goals is well-organised large datasets as well as properly selected and developed methods. It is considered that the quality of large datasets is crucial for the pace of development of existing and future ML tools in the field of architectural design.

Big data

Some remarks can be made about the 5V (Volume, Velocity, Variety, Veracity, Value) characteristics that are required for the effectiveness of large data sets in the context of architectural data sets (Beyer & Laney., 2012). In terms of volume, although the quantitative density of architectural data seems to be an advantage, the variety of types and levels of the data density is a drawback. The lack of standards similar to those used in digital images limits the possibility of expanding useful data set.

When examined in terms of speed, it is possible to consider the word from two different perspectives. The rate of increase in digital architectural data is positive in terms of creating data sets. The multidimensional structure of architectural data, due to its nature, challenges ML tools in terms of the speed at which data can be processed. There is no problem in terms of diversity. Architectural data has rich diversity in terms of content and size but the format and data availability are the major challenges. In order to comment on the accuracy and value of the data, it is necessary to examine the data sets specifically. In this respect, comparison of synthetic data sets with real data and their usefulness can be discussed. Based on these 5 characteristics, it can be seen that architecture data sets are particularly sensitive to format, scale, and detail.

Besides the 5V's, in terms of big data architecture, requirements scalability, high performance computing, modularity, consistency, interoperability and usability are also important factors that affect the performance of the generated data sets (Kaisler et al., 2013). The methods and approaches suggested in this study have been developed to consider the factors mentioned above (Chen et al., 2015).

Existing datasets

Classical design presentations are two-dimensional (2D) projections and are the most typical method of architectural visualization. Besides, three-dimensional (3D) physical models, 3D drawings are also presented in 2D. That's why most architectural databases are 2D (Nauata et al., 2020).

2D datasets are easily expandable and can be combined to create large and extensive datasets even if they are of different size, detail and type. A large number of such datasets are designed to support the implementation of various AI tools in the field of architectural practice with a variety of elements and labelling systems (Brown, L.,et al 2020). It is important to note that training ML models only on 2D data can neglect some important spatial relationships.

The available 3D architectural datasets are examined and they are seen to vary in their approach. The simplicity of the structure of large datasets is due to the density and complexity of the 3D data, but few detailed datasets are available (Selvaraju,P., et al 2021). ML studies in architecture needs a 3D dataset that contains information from different phases of the design process in a consistent way. The enhancement of the usage of ML in architectural design will be a significant leap towards the creation of a single extensive 3D dataset of various problems, scales, and stages (Chang et al., 2021).

Objective of the study

This study aims to develop methods to accelerate the transformation of data of certain sizes in order to contribute to the creation of unified and coherent data sets that can meet the changing data needs of machine learning techniques in the field of architectural design. The requirements for such a dataset have been determined:

- The ability to store 3D volumetric geometric data.

- Applicability to different design phases, different levels of detail and scales,
- The ability to store a wide range of architectural elements, their corresponding data and spatial information

Best source of data to populate such a dataset is clearly 3D Building Information Modeling (BIM) data. The collection and compilation of the geometric and spatial data of BIM, as well as the architectural elements in a structured dataset would be a boost to the ML research in architectural design (Emunds et al., 2021).

There are two major tasks that need to be met in order to reach this goal:

1. Access to ready-made BIM datasets of different levels of detail and representing all functional groups;
2. Creating a data converter that would be able to translate BIM data into a tensor form according to certain parameters.

In this paper, the focus is not on the collection of BIM data, but on the conversion of BIM data into tensor data type that can be easily used in machine learning. It is aimed to design a tool that can automatically and accurately convert architectural projects completed in the BIM environment into tensor data and convert them consistently between different levels of detail, scales and design phases.

The paper focuses on the description of the method and the processes. The proposed transformation method is limited to 3D models and BIM classes obtained from BIM data. The performance analysis of the transformation method was performed on a limited number and scale of residential buildings. The maximum working area was determined as 25.6 m^3 and a maximum resolution of 1024x1024x1024 voxels.

FRAME

Volumetric data preference

Large amounts of important data are stored in architectural projects, including data on the geometric definitions of the objects, the relations between the elements, the categorical information and the spatial structures. Mesh representation is a standard approach in 3D CAD software, but its undefined vertex count and variability in architectural elements makes it incompatible with ML methods that require a structured multidimensional matrix. However, this paper explores the use of volumetric data over mesh and point cloud data for application in ML for architecture, while acknowledging a loss of precision through the use of image-based processing techniques.

For storage of volumetric data this paper uses the ‘image stack’ approach which is commonly employed in medical imaging (CT, MRI) and engineering studies especially in microscopic analysis (Mathie and Strickland.,1997). Where physical structures are sliced in medical and engineering contexts, this study creates volumetric data by slicing digital twins of architectural structures within the BIM environment.

The key reasons for applying the Image Stack method are as follows:

- The potential of using pre-trained models and ML approaches for image stack processing in medical and engineering fields (Chambon et al., 2022).
- The existence of ready-to-use Image Stack visualization and analysis tools for medical imaging applications (Pieper et al., 2004).
- The capability of proposed data conversion approach, based on digital slicing, to be faster and more versatile than conventional voxel-based converters when used for various scale and resolution projects.

- The ability of the Image Stack method to collect data at different resolutions along the Z-axis (the vertical axis) to improve data density and to create a good basis for the development of automated completion methods.

Based on these considerations, the data converter proposed for this study has been developed using the Image Stack framework.

BIM file formats and limits

This study remaps class structures from Autodesk Revit, Graphisoft Archicad, and IFC file formats which are the most popular ones used in BIM workflows. The BIM classes are not the complete representation of the BIM data but the use of BIM classes is very essential in the definition of the architectural description along with the geometry. The data conversion experiments were performed using the softwares, Autodesk Revit and Rhino Inside Revit plugins. Despite the fact that the present study is based on BIM data conversion, the proposed method can also be used for defining architectural meshes where BIM classes are assigned manually using the data converter.

The proposed image stack conversion method has no theoretical resolution limits but practical resolution limits were based on hardware limitations and requirements of ML tools. The tests were performed from 32x32×32 px to 1024×1024x1024 px and all the images were saved in a 4-channel (RGB-A) format to incorporate the non-geometric information.

The study's experimental volume was a 2560 cm^3. At 1024×1024 px resolution the edge of the unit cube is 2.5 cm and at this resolution it captures details at the level of a 1/50 scale drawing. At 32×32 px the edge of the unit cube is 80 cm and can be used for massing and spatial relationships. The resolution of 256×256 px (10 cm edge of the unit cube) was identified as optimal as it ensures that there is enough detail without increasing the tensor dimensions to the point where it becomes infeasible to use them on consumer hardware.

Target areas

Target users of the data converter developed in this study are ML and AI researchers in the Architecture, Engineering, and Construction (AEC) and BIM domains. The efficiency and adaptability that the converter offers across the various stages of design are believed to be useful in the generation of datasets. More importantly, it is anticipated that, in the specified format, the converter will assist in data sharing and collaborative research among researchers applying different approaches. As a result, because of the contained element information and the geometric data relationships of the BIM-based volumetric datasets, it is expected that the research conducted using methods such as 3DGAN, 3DVAE, and 3D Diffusion Models will diversify. In addition, vertical resolution flexibility of the image stack method may help ML approaches for auto-completion. It is expected that these methods can be used in the field of Heritage Building Information Modeling (HBIM) (Panayiotou & Kontovourkis., 2024).

3D BIM DATA - IMAGE STACK CONVERTING METHOD

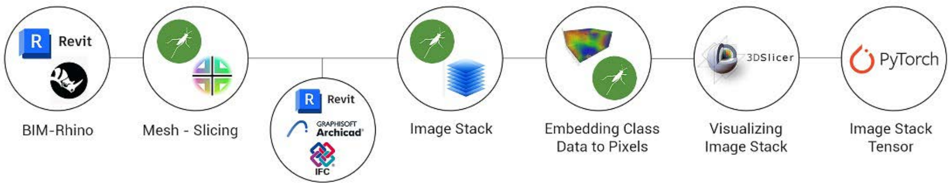
The workflow and the recommended usage process of the data converter that involves collecting 3D BIM data, converting it to tensor data within certain boundaries, using it with ML tools, and then decoding the results back to 3D BIM data can be explained in 10 steps (Figure 1).

1. BIM Data Collection
2. Choosing Resolution and Embedded Data Types Based on Usage Purpose
3. Slicing of Mesh Data from BIM at the Set Resolution
4. Embedding BIM Class Information into the Sliced Geometric Data Using Colour Codes

5. Conversion of Vectors to Images and Creating Image Stack
6. Conversion of the Image Stack Data into Tensor and Further Processing with the Help of ML Tools
7. Reconstruction of Tensor Formed by ML Prediction Models to Image Stack Form
8. Virtual Presentation of the Information Using Image Stack Viewers to Validate the Output
9. Reading of the Data in Pixel by Pixel format from the Image Stack and Conversion to Vector Geometric Data
10. Assigning BIM Classes to the Generated Vector Geometries using Layers.

For the execution of the above steps, we employed Autodesk Revit, Rhino.Inside.Revit plugin, Rhinoceros, grasshopper Plugin, and the Aviary module for Grasshopper in this research.

Figure 1
Coversion Steps



Nevertheless, other software environments can also be selected for the application of these steps since the CAD operations within the steps are relatively basic and include operations such as intersection, layer assignment, and boolean operations.

Defining classes

In the proposed conversion method, it is crucial to extract class information from the input BIM data. As Revit, Archicad, and IFC have different means of defining classes, a unified, but simplified classification system with 27 classes was proposed to ensure compatibility. Architectural and structural components were described in detail, while the MEP components were categorized more generally. Figure 2 shows a 4-channel system for assigning class data to pixels in order to enable ML evaluation.

The classification uses four channels:

- Alpha Channel: This channel contains the Spatial-Elemental Classification (Spatial Data, MEP, Furniture, Architectural, Structural).
- Red Channel: Directional Classification (beams, floors, roofs = Horizontal; walls, columns = Vertical).
- Green Channel: Relations between Architectural Elements Classification (Spatial, Furniture, MEP, Embedded, Covering, Vertical Circulation, Boundary, Structural).
- Blue Channel: Distinctive Classification, to improve the distinction between elements of the same kind.

The Alpha channel defines elements relationally, and the Red channel, orientation. The Green channel assigns values to the elements according to the architectural relationships, e.g. doors and windows within walls. The Blue channel increases the separation between elements of the same kind by manipulating value spacing. For instance, partition walls and structural walls have the same value in Alpha, Red and Green channels but the structural wall is given a higher Blue value for easy distinction.

Although the produced colours may not accurately represent the element groups, analyzing channel values helps to understand class definitions and improve the interpretability of ML model.

Gathering BIM data

The BIM data was generated using Autodesk Revit and then imported into Rhinoceros with the help of the Rhino.Inside.Revit plugin. First, the imported building model was enclosed by a bounding box in Grasshopper to check the dimensions of the model. If the model is within the set limits it was positioned at the lower left corner of the working area to make sure that it is aligned during the slicing and image conversion process.

	CLASSES	ALPHA	RED	GREEN	BLUE	A-RGB
1	Coverings	190	250	120	88	
2	Ceilings	190	10	120	128	
3	Columns	190	250	200	167	
4	CurtainWalls	210	250	200	118	
5	Doors	210	250	100	59	
6	Furnitures	125	125	40	98	
7	LightingFixtures	140	125	40	29	
8	PlumbingEquipment	160	125	40	226	
9	Railings	210	10	100	186	
10	Ramps	210	80	140	196	
11	Roofs	210	80	200	49	
12	Slabs	210	10	200	255	
13	Spaces	70	125	10	235	
14	Stairs	210	80	140	69	
15	StructuralBeams	255	10	250	147	
16	StructuralColumns	255	250	250	78	
17	StructuralFoundations	255	10	250	177	
18	StructuralFloors	255	10	250	0	
19	StructuralGeneric	255	125	250	157	
20	StructuralWalls	255	250	250	245	
21	VerticalCirculation	175	250	140	137	
22	Walls	210	250	200	10	
23	Windows	210	250	100	206	
24	MechanicalGeneric	105	125	70	39	
25	ElectricalGeneric	110	125	70	216	
26	PlumbingGeneric	115	125	70	108	
27	OutofScope	0	125	10	20	

Figure 2
4-Channel
Class Codes

Then, architectural, structural, and MEP elements, and room volumes were assigned to the predefined layers based on the classification data available from Rhino.Inside.Revit. This is because classification data was easily encoded into pixels during image conversion due to this pre-assignment. The last operation was to convert all the elements to mesh data and incorporate the classification data into the layers in order to proceed to the image stack transformation.

These steps were combined into one Grasshopper definition and executed with the help of a 'create' button. The times of executions are presented in the Experiments and Results section. In addition, if these steps are performed manually, it is possible to assign mesh data from other software to the appropriate layers and use the method in other formats.

Slicing for image stack

To convert the BIM-imported model into an image stack, the resolution has to be set in order to export the model imported from BIM to the image stack. The sizes are also set in powers of two for compatibility with ML (32^3–1024^3) Users choose the resolution according to their goals.
Grasshopper executes the slicing process:

1. XY planes are evenly spaced based on the resolution.
2. Mesh elements are intersected with planes.
3. Closed planar polygons from intersections are hatched according to assigned layers and grouped.

After slicing, the hatched polygons are converted to images. It is obvious that the transition is made from raster to vector data, with accuracy being lost. The slicing stage is important because it defines how the image stack that is the output of the rendering pipeline displays various architectural elements (Figure 3).

Embedding BIM class information to pixels

After cutting the building model, it is converted from vector data to raster images. First, this is colored according to predefined presets:

- Simplified BIM Classes (BIM Elements)
- Volumes (Rooms & Boundaries)
- Structural Elements Only
- Building Mass

Other presets can be created for certain machine learning tasks.

The user then chooses an image format, and prefer 4-channel formats like.png and.tiff. Using Grasshopper's Aviary module, the 'Compose Drawing' and 'Drawing to Bitmap' tools are used to create images of slices and are saved with a user defined prefix. The last step is the conversion to a tensor. Although this can also be done in Grasshopper with add ons, this study uses PyTorch for faster processing and better GPU utilization.

Figure 3
Slicing for Image Stack

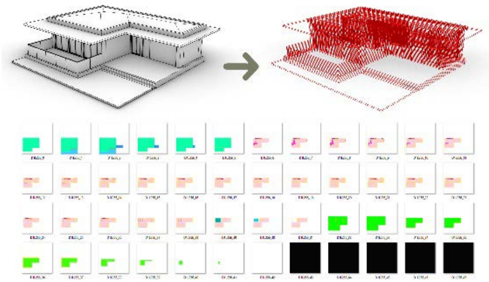


Image stack to tensor

The simplest step of the proposed method is the formatting of image stack data into tensor. Image stacks are transformed into tensors using NumPy, Skimage and PyTorch although other machine learning libraries can be used. The tensor shape is dependent on resolution; for example, a 256^3 image stack is stored as (256, 256, 256, 4), where all voxel and 4-channel data are saved. The data is saved in NumPy (.npy) or PyTorch (.pt) format. A 256^3 model can be ~65 MB in NumPy format and higher resolutions like 1024^3 will increase the file size significantly. Tensors can be saved on a per subject basis or in a single dataset. After prediction models have been processed, tensors are converted back to image stacks in the same manner. Although it is possible to directly convert to 3D voxel data, the study focuses on image stacks in order to assess their usability and performance.

Visualizing image stacks

The data is easily readable in separate layers as Image Stack in any image viewer. However, what is important for the user is to review the 3D structure of the slices in a digital environment. The tools used in the medical field for the generation of 3D representation from CT scans and MRI images can also be used for this purpose. The 3D Slicer software was tested for this purpose and success was achieved. The architectural data in the form of Image Stack with pixel data embedded in it was visualized in three dimensions. This capability is especially useful for the review of the output of the machine learning models. The data will be converted to 3D before further processing to subsequent stages, this will help in avoiding time losses. Also, it is useful for investigating large image stacks datasets.

Reconstruction of BIM data from image stacks

The process of using prediction models image stacks data to BIM models seems similar in concept but is more complicated in terms of CAD operations. The necessary steps for this transformation are listed below:

1. RGB-A values of all the pixels in each image are read.
2. Using the values and resolution, a simple mesh plane of the correct class layer is set to each pixel according to the slice height in the image stack.
3. The unit planes that are of the class in the slice are merged to get the outer contours.
4. The contours are extruded by a unit height to make them solid models.
5. All the solid models of the same class are then combined using a union operation after applying the above four steps to all layers.
6. The elements are transferred to the Revit program using Rhino.Inside, and the class information is obtained from the layer.

The reconstruction algorithm is more complex and therefore more computationally expensive than the image stack conversion algorithm. It can also be directly read from tensor data and a more efficient process can be defined through voxel-mesh converters.

EXPERIMENTS & RESULTS

The performance of the proposed converter has been analyzed in two main aspects:

1. The efficiency of the converter in transforming BIM data into an Image Stack, evaluated in terms of processing time and loss of detail.
2. The usability of the converted data in machine learning applications.

For these experiments, 20 different residential buildings of varying sizes, modeled in Autodesk Revit, were utilized. Five of these structures correspond to realized architectural projects, while the remaining 15 were specifically designed for this study to assess the converter's performance across diverse geometric configurations (Figure 4).

Converter performance tests

The performance of the converter has been evaluated using the smallest and largest structures, as well as the average performance across the remaining models. The results are presented in Table 1.

Each project was converted at resolutions of 64^3, 256^3, and 1024^3. The parameter V_1 represents the volume of the bounding box of the structure in cubic meters, while V2 indicates the volume of the structure as a percentage of the total workspace.

ObjC denotes the total number of elements in the building model, including room meshes. T_1 refers to the transfer time of BIM data to the Rhino environment, to the Rhino environment, T_2 represents the time required for the conversion of BIM data into the Image Stack format, and T_3 measures the time taken to convert Image Stack data back into the Rhino environment.

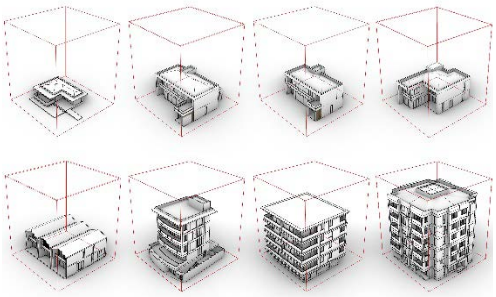


Figure 4
Examples of Models Used in Tests

The test results indicate that, aside from resolution, the number of elements in the building significantly influences the conversion time to an Image Stack. However, when converting from the Image Stack back to a mesh format, resolution emerges as the sole determining factor affecting processing time. To compare the speed of the Image Stack converter with an alternative voxel-based method, the same models were converted using the VoxelTools module in Grasshopper at identical resolutions. The comparative time measurements are presented in Table 2.

When converting a vector-based representation into a raster structure, deviations are unavoidable. The extent of these deviations depends on the conversion resolution and the

	V_1 (m^3)	V_2 (%)	ObjC	T_1 (s)	T_2 (s)	T_3 (s)
S.64^3	163,6	0,9	32	2	>1	21
A.64^3	3827,2	22,8	827	5	3,5	25
L.64^3	12364	73,6	2128	8	5	27
S.256^3	163,6	0,9	32	2	2	202
A.256^3	3827,2	22,8	827	5	14	202
L.256^3	12364	73,6	2128	8	18	204
S.1024^3	163,6	0,9	32	2	29	972
A.1024^3	3827,2	22,8	827	5	82	972
L.1024^3	12364	73,6	2128	8	212	978

Table 1
Converter Performance

proportions of the structural elements. For instance, in a 256^3 resolution, buildings composed of elements with 10 cm thickness exhibit minimal deviation.

Table 2
Converter Comparison

	Image Stack Converter (s)	VoxelTools (s)
Small 256^3	2,1	32,8
Avarage 256^3	14,4	Min. 1278
Large 256^3	17,4	N/A (Crash)

Performance of image stack in ML

As this research is mainly focused on BIM-to-image stack conversion, a thorough investigation of machine learning models was not carried out. But to check the effectiveness of the produced image stacks, small scale experiments were conducted.

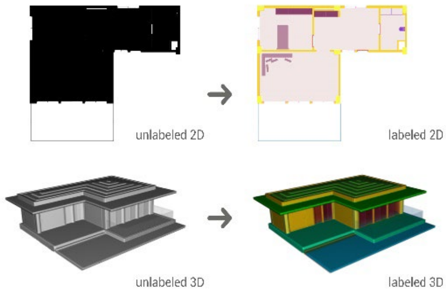


Figure 5
Segmentation Examples

Experiments of semantic segmentation were performed using 256^3 resolution samples created by the proposed converter, and both 2D and 3D U-Net architectures were employed. The 2D segmentation was more accurate as a result of the increased number of training samples, but the 3D segmentation was not as efficient as that. (Figure 5)

Two more experiments were carried out: 3D upscaling autoencoder and denoising autoencoder. The first experiment was to predict high resolution image stacks from low resolution ones, to see whether low detail early stage design models could be refined into detailed BIM models. The second experiment was to complete the task of reconstructing high resolution image stacks from noisy, low resolution inputs to see if parts that were missing could be accurately inferred. Since the dataset was small, both in the case of upscaling and denoising, the results were not satisfactory.

CONCLUSIONS

In this paper, a conversion method of arranging BIM data in the form of image stacks for applications in ML was suggested. The methodology and its performance were elaborated throughout the paper. The proposed transformation method is limited to 3D models and BIM classes extracted from BIM data. A class simplification is also proposed in the paper.

Small dataset of 20 buildings is used to analyse the performance of the method and its outputs. Although the dataset has a limited sample size and single building function, the samples are selected in different sizes and typologies.

The method proposed and the data standardization it aims to achieve are expected to be useful to ML researchers in the area of architectural design. Several ML models such as 3D GAN, D VAE, and 3D diffusion models have been applied to architectural design. These methods are costly in terms of computational time and requirements. Different data collection and preparation methods can help in their creation. In addition, the standardized format of the dataset enables the sequential use of various ML techniques. For instance, a 3D VAE model can generate massing options, which can then be used to predict architectural details given the initial massing outputs by a denoising autoencoder.

Although raster based data formats such as image stacks and voxels are not as accurate as vector data, the rate of advancement in computation is largely overcoming the problems associated with resolution. Consequently, the image stack, with its easy-to-use format, has many potential future uses.

Future experiments with the larger datasets will reveal the limitations and enhance the classification and efficiency of the algorithm of the method. Moreover, further optimization by professional developers could improve the performance and implementation of the conversion process proposed in the paper.

REFERENCES

Topuz, B. and Alp, N.C. (2023) 'Machine learning in architecture', *Automation in Construction*, 154, 105012.

Beyer, M.A. and Laney, D. (2012) *The importance of 'big data': A definition*. Stamford, CT: Gartner.

Kaisler, S., Armour, F., Espinosa, J.A. and Money, W. (2013) 'Big data: Issues and challenges moving forward', in *46th Hawaii International Conference on System Sciences (HICSS)*, 7–10 January, Wailea, HI, USA. IEEE, pp. 995–1004.

Chen, H.M., Kazman, R., Haziyev, S. and Hrytsay, O. (2015) 'Big data system development: An embedded case study with a global outsourcing firm', in *1st IEEE/ACM International Workshop on Big Data Software Engineering (BIGDSE)*, 16–24 May, Florence, Italy. IEEE, pp. 44–50.

Nauata, N., Chang, K.H., Cheng, C.Y., Mori, G. and Furukawa, Y. (2020) 'House-GAN: Relational generative adversarial networks for graph-constrained house layout generation' in *Computer Vision – ECCV 2020: 16th European Conference*, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16. Cham: Springer, pp. 162–177.

Brown, L., Yip, M., Gardner, N., Haeusler, H., Khean, N., Zavoleas, Y. and Ramos, C. (2020) 'Drawing recognition—Integrating machine learning systems into architectural design workflows', in *Proceedings of the 38th International Conference on Education and Research in Computer Aided Architectural Design in Europe (eCAADe)*, Vol. 2, pp. 289–298.

Selvaraju, P., Nabail, M., Loizou, M., Maslioukova, M., Averkiou, M., Andreou, A. and Kalogerakis, E. (2021) 'BuildingNet: Learning to label 3D buildings', in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10397–10407.

Chang, K.H., Cheng, C.Y., Luo, J., Murata, S., Nourbakhsh, M. and Tsuji, Y. (2021) 'Building-GAN: Graph-conditioned architectural volumetric design generation', in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 11956–11965.

Emunds, C., Pauen, N., Richter, V., Frisch, J. and van Treeck, C. (2021) 'IFCNet: A benchmark dataset for IFC entity classification', in *EG-ICE 2021 Workshop on Intelligent Computing in Engineering*, Vol. 166. Universitätsverlag der TU Berlin.

Mathie, A.G. and Strickland, N.H. (1997) 'Interpretation of CT scans with PACS image display in stack mode', *Radiology*, 203(1), pp. 207–209.

Chambon, P., Bluethgen, C., Langlotz, C.P. and Chaudhari, A. (2022) 'Adapting pretrained vision-language foundational models to medical imaging domains', *arXiv preprint*, arXiv:2210.04133.

Pieper, S., Halle, M. and Kikinis, R. (2004) '3D Slicer', in *2004 2nd IEEE International Symposium on Biomedical Imaging: Nano to Macro*, 15–18 April, Arlington, VA, USA. IEEE, pp. 632–635.

Panayiotou, P.N. and Kontovourkis, O. (2024) 'A holistic documentation and analysis of timber roof structures in heritage buildings using scan to HBIM approaches' in *Proceedings of the 42nd International Conference on Education and Research in Computer Aided Architectural Design in Europe (eCAADe)*, Vol. 1, pp. 715–724.