Eurographics Workshop on 3D Object Retrieval (2021)
S. Biasotti, R. M. Dyke, Y. Lai, P. L. Rosin, and R C. Veltkamp (Editors)

Shape Classification of Building Information Models using Neural Networks

I. Evangelou[1] and N. Vitsas[1] and G. Papaioannou[1] and M. Georgioudakis[2] and A. Chatzisymeon[2]

[1]Department of Informatics, Athens University of Economics and Business, Greece
[2] Nomitech Ltd. UK

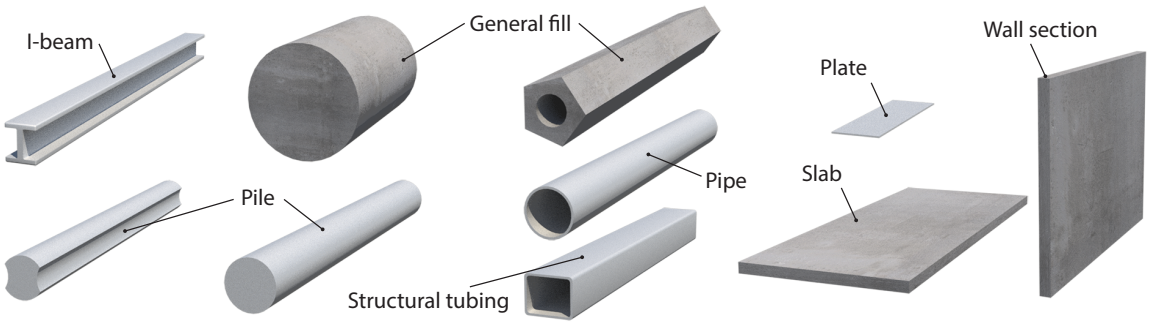

Figure 1: *Classification examples for typically problematic BIM element cases.*

Abstract

The Building Information Modelling (BIM) procedure introduces specifications and data exchange formats widely used by the construction industry to describe functional and geometric elements of building structures in the design, planning, cost estimation and construction phases of large civil engineering projects. In this paper we explain how to apply a modern, low-parameter, neural-network-based classification solution to the automatic geometric BIM element labeling, which is becoming an increasingly important task in software solutions for the construction industry. The network is designed so that it extracts features regarding general shape, scale and aspect ratio of each BIM element and be extremely fast during training and prediction. We evaluate our network architecture on a real BIM dataset and showcase accuracy that is difficult to achieve with a generic 3D shape classification network.

CCS Concepts

• ***Computing methodologies*** → ***Neural networks; Shape analysis;***

1. Introduction

Building Information Modelling consists of processes supported by specialised software suites and tools, which aim to digitally describe functional and physical entities of large construction projects. *Building Information Models* (BIM) adhere to standardised specifications and open or proprietary data exchange formats, widely used by the construction industry, to support the full lifecycle of civil engineering projects from design, planning and costing to the construction and even operational stage. A BIM is a hierarchical collection of individual elements characterised and often labelled by role and function and accompanied by detailed textual and numerical data describing measurements, quantities, materials, vendors and other type-specific attributes.

BIMs are nowadays processed and analysed by specialised software to (semi-) automatically estimate construction costs, quantities and validate digital plans. These processes require that a meticulous annotation of BIM elements has been performed during the creation of the digital model. However, more often than not, elements in BIM files are incompletely labelled, use different conventions or are misclassified. Textual information can sometimes be used to narrow down the possible classification labels but is typically unreliable. Therefore, the need arises for shape-based com-

© 2021 The Author(s)
Eurographics Proceedings © 2021 The Eurographics Association.

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

DOI: 10.2312/3dor.20211306

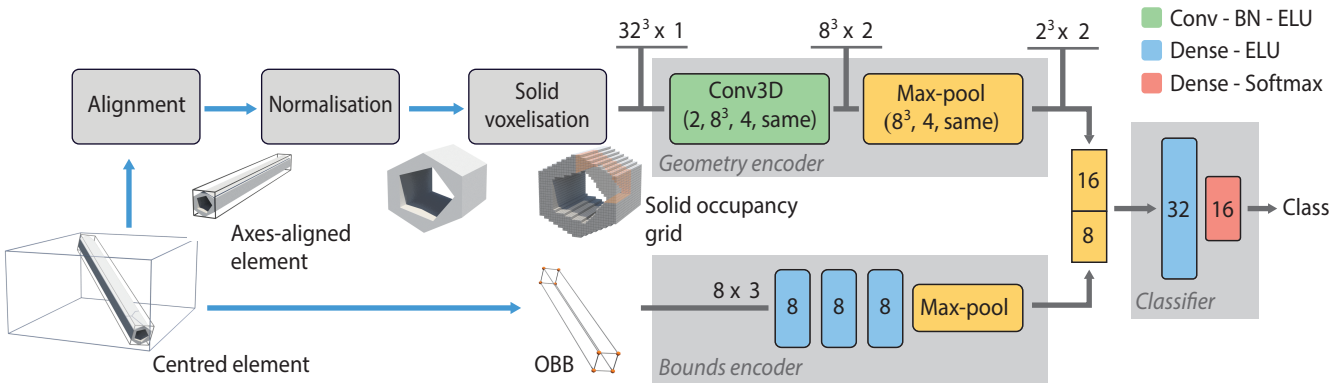

Figure 2: *The dual-network architecture of our BIM element classification module. Numbers in the dense layers and output embeddings indicate feature dimensions.*

putational methods for automatic labelling that can assist in the tedious task of BIM data cleanup and consolidation.

In this paper we present a modern, neural-network-based classification solution to automatic BIM element labelling, based only on geometric information. Such a solution can effectively label many important structural elements in a BIM file and provide meaningful suggestions during BIM data processing. The classification of BIM elements requires careful handling and cannot be directly addressed by an off-the-shelf application of a widely-used neural network. Firstly, many element classes are geometrically quite similar and require special sampling and feature extraction to distinguish. Additionally, elements come in many form factors and sizes, thus requiring careful normalisation of their bounds. Finally, many elements differ only in absolute scale and typical orientation. Therefore, the architecture should not be completely scale and rotation invariant, but at the same time it should accommodate different poses for many categories of elements. For example, as shown in Fig. 1, a simple box can be a (steel) plate, a concrete slab or a wall section and the only practical, geometric discrimination factors are its absolute scale and minor axis alignment.

Our task falls directly under the general problem of 3D shape classification, which has been extensively researched over the past. Hand-crafted 3D shape descriptors have been used by researchers in 3D search engines and sketch-based modelling systems. The interested reader may refer to [KYZ13] for a comprehensive study of traditional 3D shape descriptors. However, state-of-the-art results have been achieved recently using deep learning methods. Several data representations have been proposed as input to neural network architectures. These include voxel grids [MS15], point-clouds [CSKG17], rendered images [SMKL15], hierarchical structures [RUG17] and raw primitive data [FFY*19]. Voxel grids are very efficient to store, manipulate and access and lend themselves nicely to modern network architectures that use convolutional layers. VoxNet [MS15], was one of the first voxel-based network architectures to demonstrate great performance in terms of accuracy, memory and evaluation time. Surface samples are taken on a 32^3 regular grid and are used as input in a series of convolutional layers followed by a dense layer for class inference.

Although a dense, hierarchical classifier could also be used for this task, we specifically desired a lean architecture that would enable the classifier to run on typical low- to mid-range desktop or laptop machines and workstations, thus enabling the integration in civil engineering software, without introducing extraneous requirements. Furthermore, we expressly needed to take into account the scale and pose of certain object types, which warrants a specialised architecture for BIM classification. Traditional classifiers, based on hand-crafted features, were also initially considered, but these would require significant effort from applied field experts, and potentially higher computational cost, without any clear indication of performance benefits.

2. Method Description

Our BIM element classification module processes triangular meshes and is jointly trained on two separate neural branches dedicated to extract meaningful features from the input (see also Fig. 2). The number of BIM element classes within a certain construction category, associated with a plan layer, is not very large, but the elements can be structurally similar. Element dimensions have significant impact on the class selection and our classification architecture takes them into account alongside the shape of the input geometry, as described below. The first sub-network consumes the voxelised representation of the element and proceeds with traditional convolutional layers, whereas the second one directly routes the eight associated tight oriented bounding box corners to a point-based network architecture. Carefully transforming the input for each sub-network, based on trivial-to-implement procedures, can effectively boost the discrimination capabilities of our architecture and resolve hard to classify cases, which regularly emerge in this context.

Geometry encoder. This path processes the triangular mesh input and results in a feature vector, which is propagated to the classifier after concatenation with the encoded bounds result of the architecture. The mesh is centred and aligned with the reference frame axes, after applying the inverse oriented bounding box rigid transformation matrix. Next, to handle objects within the same class but of different proportions, we normalise each dimension separately.

© 2021 The Author(s)
Eurographics Proceedings © 2021 The Eurographics Association.

Note that this normalisation approach is in contrast to the standard practice, which is to uniformly scale the object according to the maximum dimension, retaining object proportions. Our approach maximises voxel sample density and allows a descriptive representation even for small voxel grids. In the case of BIMs, this is important, since it allows the discrimination between solid and hollow elongated parts (e.g. solid beams and piles versus pipes and other tubular structures).

Next, the normalised meshes are solidly *voxelised* into a binary occupancy grid, which constitutes the input of this branch of the neural network. The most standard volumetric representation for triangular meshes is that of surface voxelisation, as used by VoxNet. A region enclosing the entire element is regularly sampled and a binary occupancy grid is constructed marking voxels intersected by the mesh surface. However, to be able to discriminate elements with similar hulls but different internal structure, such as pipes and cylindrical piles (see Fig. 1), as is the case in BIM classification, we opted to perform solid voxelisation. Despite being a significantly more involved procedure, all voxels inside the mesh are marked as occupied, in addition to shell samples. We use a conservative, point-in-mesh query, which determines the signed distance of each voxel centre to the closest mesh point. This approach allows us to robustly handle elements that have holes or disjoint vertices.

The 32^3 binary occupancy values of the voxel grid are fed to convolutional sub-sampling layers, which generate the latent encoding of each element. Due to the relatively small number of meaningful classes in each contextually separate information layer for buildings, network topology and the number of trainable parameters can be kept small, which is in line with the practical requirements of element processing; applications running on low- to mid-range devices usually have to analyse thousands of elements per scene.

Bounds encoder. Performing coordinate normalisation independently across dimensions may improve descriptiveness among several classes (e.g. L-beams versus U-beams) but dropping features such as absolute scale and proportions destroys critical discriminative evidence among them. One simple and clear example of this is the distinction between a slab and a simple wall section. After alignment and normalisation both are mapped to the unit cube, making them indistinguishable by the geometry encoder features alone.

The bounds encoder in our pipeline, jointly trained along with the geometry path, is responsible for resolving these ambiguities. After centring a BIM element and calculating its oriented bounding box, the eight box corners are extracted and their coordinates are converted to metres. These are then passed through the second network and latent encoded features are obtained via a typical point-based architecture, similar to PointNet [CSKG17].

Classifier. The output embeddings from the geometry and bounds encoders, with dimensionality of 16 and 8 respectively, are concatenated to form the final shape descriptor. This vector is then forwarded to a dense layer followed by a softmax layer, producing the predicted classification labels. Jointly optimising our network without auxiliary hand-crafted features, forces the network to learn meaningful representations from each branch conditioned on the input elements.

3. Evaluation

For all our experiments we trained the network for a maximum of 50 epochs with early stopping monitored based on 0.02 absolute difference for validation loss and a patience of 5 epochs. Upon early termination, the best model in terms of validation loss is recorded. We additionally use the default Adam optimiser and a batch size of 32. The discussed architecture in Section 2, has a total of 2534 trainable parameters. Our Tensorflow implementation and the models discussed in the following sections are publicly available at: https://github.com/cgaueb/deep_bim.

Dataset. Our dataset consists of 16 BIM classes which are roughly balanced and result in 853 elements. We further augment every element with random rotations around the up direction, reflecting plausible orientations in a construction, with the exception of the Beam classes, which are freely and arbitrarily oriented, as in reality. This results in 2571 elements for training and testing. An overview of the training and test classes is presented in Figure 3. The number of classes, although limited, reflects important building elements in this particular scenario, in terms of BIM analysis and cost estimation. Each BIM model and information layer typically uses a subset of all meaningful BIM element categories and therefore makes more sense to build and train multiple, leaner classifiers rather a single, all-encompassing system.

Experiments. To demonstrate the expressiveness of our architecture, we evaluated our model against two baseline architectures. The first (Baseline 1) is a traditional voxel-based approach, which generates an occupancy grid for the solid voxelisation using the standard normalisation technique with no alignment. The second (Baseline 2) uses the same voxelisation scheme, but introduces the alignment and normalisation method proposed in Section 2. To make the comparison fair, we increased the number of parameters for the Baseline models to match the learning capacity of our architecture. In all our experiments we use a 70/30, train/test split.

In Table 3 we report per class accuracy for a subset of indicative example cases that highlight the weaknesses of the baseline methods versus our approach, which retains class separability by construction. Starting with the I-beam and U-beam cases we can observe that standard voxelisation (Baseline 1, second column) is ineffective for these cases. This error manifests from the fact that elongated meshes tend to generate thin voxel representations effectively eliminating surface details. An obvious solution to the latter, would be to greatly increase the voxel grid size. This, however, would result in an excessive number of trainable parameters that is undesirable and can be avoided by employing the Baseline 2 model (third column). The uniform normalisation and alignment resolve such problems but a number of other cases become infeasible to classify correctly. This is evident for instance with classes such as Plate and Slab since the Baseline 2 network gets an almost identical input for both classes. On the other hand, conditioning the network classifier with the bounds encoder, most information about the original shape is retained and class separability is achieved, as demonstrated by the results in the fourth column. For completeness, we also record in Table 3, the overall accuracy and number of trainable parameters for each model and in Figure 3 the confusion matrices of our method versus the Baseline 1 model in the test subset.

© 2021 The Author(s)
Eurographics Proceedings © 2021 The Eurographics Association.

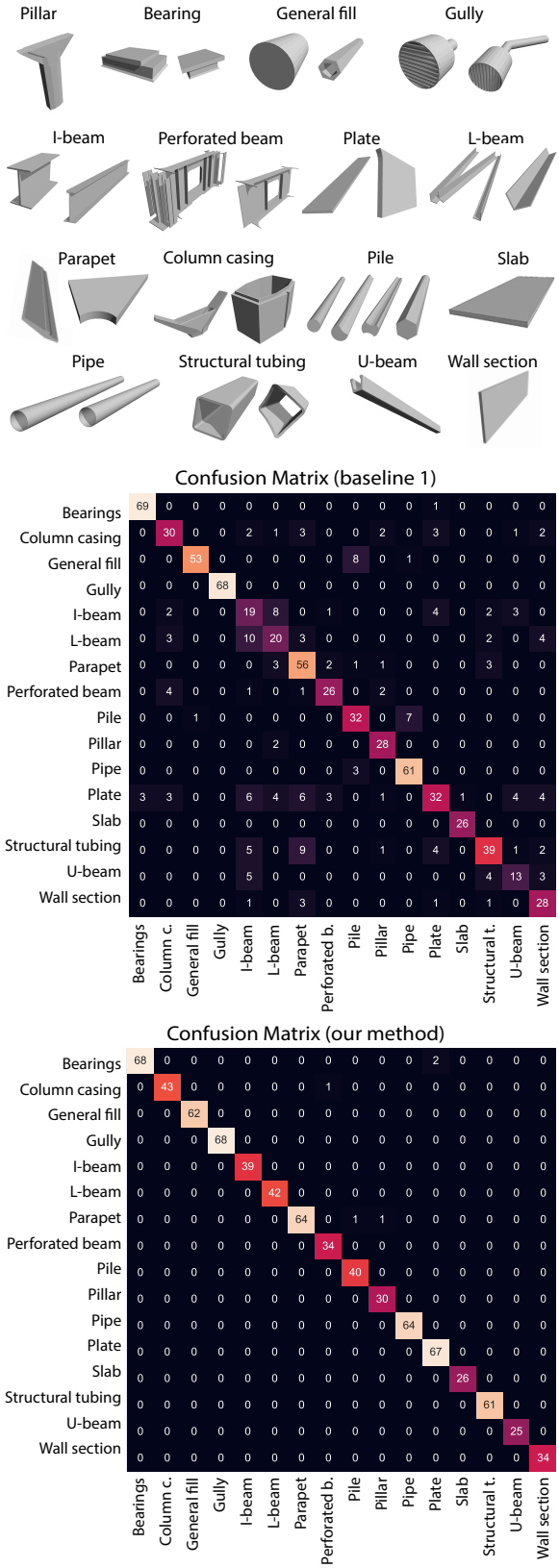

Figure 3: *Classification performance for each class (top). Confusion matrices for the test subset are for Baseline 1 implementation (no alignment, uniform normalisation) and our method.*

	Accuracy per model (train / test)		
Class name	**Baseline 1**	**Baseline 2**	**Our method**
I-beam	0.82 / 0.48	1.0 / 1.0	1.0 / 1.0
U-beam	0.50 / 0.52	1.0 / 1.0	1.0 / 1.0
Gully	1.0 / 1.0	1.0 / 1.0	1.0 / 1.0
Bearings	1.0 / 0.98	0.97 / 0.97	1.0 / 1.0
Plate	0.59 / 0.47	0.54 / 0.67	1.0 / 1.0
Slab	1.0 / 1.0	0.57 / 0.53	1.0 / 1.0
overall acc.	0.84 / 0.76	0.92 / 0.93	0.99 / 0.99
#params	2873	2873	2534

Table 1: *Reported accuracy on training (left) and test (right) sets for a subset of indicative classes, see also Figure 3. Last two rows report mean overall accuracy over the whole dataset along with the capacity of each model.*

4. Discussion and Conclusions

We presented a network architecture that performs reliably in compact collections of hard-to-discriminate parts with many variations, typically found in BIM datasets. The network was successfully deployed and integrated as a shape classification module into *CostOS*, a cost estimation software developed by Nomitech Ltd. running in real-time on a wide range of devices. Our results strongly support and suggest the applicability and effectiveness of the approach to a wider range of elements and classes. The dual-network design of the network allowed us to keep the number of learnable parameters at a minimum. This leaves room for a substantial increase in learnable parameters and in the future, we want to evaluate our network architecture on larger and more detailed datasets.

References

[CSKG17] CHARLES R. Q., SU H., KAICHUN M., GUIBAS L. J.: Pointnet: Deep learning on point sets for 3d classification and segmentation. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 77–85. doi:10.1109/CVPR.2017.16. 2, 3

[FFY*19] FENG Y., FENG Y., YOU H., ZHAO X., GAO Y.: Meshnet: Mesh neural network for 3d shape representation. *Proceedings of the AAAI Conference on Artificial Intelligence 33*, 01 (Jul. 2019), 8279–8286. doi:10.1609/aaai.v33i01.33018279. 2

[KYZ13] KAZMI I. K., YOU L., ZHANG J. J.: A survey of 2d and 3d shape descriptors. In *10th International Conference Computer Graphics, Imaging and Visualization* (2013), pp. 1–10. doi:10.1109/CGIV.2013.11. 2

[MS15] MATURANA D., SCHERER S.: Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (2015), pp. 922–928. doi:10.1109/IROS.2015.7353481. 2

[RUG17] RIEGLER G., ULUSOY A. O., GEIGER A.: Octnet: Learning deep 3d representations at high resolutions. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2017), pp. 6620–6629. doi:10.1109/CVPR.2017.701. 2

[SMKL15] SU H., MAJI S., KALOGERAKIS E., LEARNED-MILLER E. G.: Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV* (2015). 2

© 2021 The Author(s)
Eurographics Proceedings © 2021 The Eurographics Association.