Highlights

Exploiting BIM objects for synthetic data generation towards indoor point classification using deep learning

Ernesto Frías, José Pinto, Ricardo Sousa, Henrique Lorenzo, Lucía Díaz-Vilariño

- a highly automated algorithm to transform BIM objects into customized point clouds
- a flexible deep learning framework to evaluate the generated synthetic data
- a comparison of classification performance using orthogonal and perspective projections
- an evaluation of greyscale images with surface curvature for classification

Exploiting BIM objects for synthetic data generation towards indoor point classification using deep learning

Ernesto Frías[a], José Pinto[b], Ricardo Sousa[b], Henrique Lorenzo[a], Lucía Díaz-Vilariño[a,*]

[a] *Universidade de Vigo. CINTECX, Applied Geotechnologies Research Group, Campus universitario de Vigo, As Lagoas, Marcosende, Vigo, 36310, Spain*
[b] *LIAAD, INESC TEC, Campus Faculty of Engineering, University of Porto, Dr. Roberto Frias, Porto, 4200-465, Portugal*

Abstract

Advances in technology are leading to more and more devices integrating sensors capable of acquiring data in a very fast and with high accuracy. Point clouds are no exception. Therefore, the large amount of available lidar data is arising the community interest by point cloud classification using artificial intelligence. Nevertheless, point cloud labelling is a time-consuming task. Hence the amount of labelled data is scarce yet. Data synthesis is gaining attention as an alternative to increase the volume of classified data. At the same time, the amount of BIM models provided by manufacturers on website databases is being increased. In line with these trends, a method to generate classified point cloud data from BIM objects is presented. The method starts by transforming BIM objects into point clouds deriving a dataset consisting of 21 object classes characterised with various perturbation patterns. Then, the dataset is split into four subsets to carry out the evaluation of synthetic data on the implemented flexible 2D deep neural framework. In the latter, binary or greyscale images can be generated from point clouds by both orthographic or perspective projection to feed the network. Moreover, surface variation feature was computed in order to aggregate more geometric information to images and to evaluate how it influences the object classification. The overall accuracy is over 85% in all tests when orthographic images are used. Also, the use of greyscale images representing surface variation improves performance

*Corresponding author
Email address: lucia@uvigo.es (Lucía Díaz-Vilariño)

in almost all tests although the computation of this feature may not be robust in point clouds with complex geometry or perturbations.

Keywords: point clouds, deep learning, BIM, object classification, data augmentation, transfer learning

1. Introduction

In the last decade, the use of 3D sensors such as laser scanners or depth cameras has been consolidated in Architecture, Engineering and Construction (AEC) industry and robotics providing a greater availability of 3D models of indoor scenes in the point cloud form (Khoshelham et al., 2017). More recently, mobile devices widely used by the population such as the ipad pro or the iphone 12 pro have integrated lidar sensors extending the possibility of acquiring point cloud data on a massive scale. In both computer vision and 3D modelling, point cloud classification is an active topic because it is a fundamental problem for the understanding of 3D scenes in the real-world. Beyond geometric analysis, traditional machine learning techniques have been applied to address the 3D classification problem (Weiss et al., 2010; Park and Guldmann, 2019). However, the successful results achieved with deep learning techniques in 2D image classification are leading researchers to adopt this approach in 3D classification (Griffiths and Boehm, 2019a; Jaritz et al., 2019).

Traditional machine learning techniques are based on teaching machines to identify patterns and extract features from data that are not perceived by humans due to the large volume and complexity of the information to be processed (Dey, 2016). Thus, the performance of these methods is strongly dependent on the design of a feature extractor that requires a comprehensive knowledge in the domain of application. This limitation has been outperformed by the newest machine-learning techniques since their capacity to interpret raw data without relying on human skills (LeCun et al., 2015).

In the recent years, Deep Neural Networks have demonstrated a high performance in applications such as speech recognition (Abdel-Hamid et al., 2012) and image recognition (Krizhevsky et al., 2012) becoming the state-of-the-art so far for both areas (Szegedy et al., 2014). Convolutional Neural Network (CNN) architecture is based on visual perception, hence its ability to interpret the nature of images which makes CNNs suitable for image classification. The requirement to have a large classified dataset available is a

major drawback of using CNNs because if labelled data are short, the deep learning model will most likely cause overfitting (Liu et al., 2016).

In many areas, the amount of classified data is not enough to train a CNN deriving a well-fitting model. This lack is commonly addressed by using data augmentation techniques which consist in applying certain transformations on the initial dataset to expand input data (Perez and Wang, 2017). Different transformations can be applied to both synthetic and real data. Data augmentation from real data can be addressed by data wrapping and synthetic over-sampling approaches depending on whether the transformations are applied in the data or in the feature-space respectively. To carry out synthetic-based data augmentation, new samples are artificially generated to be added to initial dataset (Wong et al., 2016). In this way, previous works have demonstrated that the over-sampling generation of the minority classes in imbalanced datasets can improve classifier performance (Bowyer et al., 2011).

An additional alternative commonly used to deal with inadequacy of labelled data is to reuse models already trained in another application domain or for a different task but keeping some sense of relationship. This approach, widely known as transfer learning, leverages the knowledge acquired by training a model with a large amount of data to another domain of interest with shorter data set (Pan and Yang, 2010). Network-based deep transfer learning consists in using a partially pre-trained network with a large volume of labelled data in a smaller dataset that keeps certain similarity with the data used to train the original network (Tan et al., 2018).

In last years, the development of applications using point clouds to generate as-build models from existing buildings has grown in the AEC industry (Bosché et al., 2013). However, although more and more point clouds are available, its labelling is an arduous, time-consuming and error-prone task. The use the BIM models for ongoing building monitoring has provided considerable improvements in processes of construction control (Wang et al., 2014), building energy analysis (Abanda and Byers, 2016), project documentation and coordination (Broquetas et al., 2013). Also, BIM models of objects such as pieces of furniture are provided by manufacturers and they can be retrieved from website databases. Since BIM models not only represent geometry of building components or objects but also provide semantic and functional information, they can be used as classified 3D models. Recently, building BIM models have been used to generate both 2D and 3D synthetic data for image classification and point cloud semantic segmentation

(Ma et al., 2020; Alawadhi and Yan, 2021). Nevertheless, training a neural network with synthetic point clouds provides models that do not generalise well with real-world data (Uy et al., 2019). To address with this shortcoming, synthetic point clouds can be perturbed by adding intrinsic undesired defects such as noise or occlusions.

This work proposes a method that explores the use of BIM object models for generating synthetic data sets composed of multiple classes including those that are not normally present in available public classified datasets. Synthetic perturbed point clouds with noise and occlusions are also created to reduce the impact of bad generalisation of classification models trained with no real data. The main contributions provided by this work are as follows:

- an algorithm to transform BIM objects into customized point clouds. Beyond data provided, the procedure has the advantages that is highly automated the minimal user intervention once BIM objects have been downloaded.
- a flexible deep learning framework to evaluate the generated synthetic data. This also implements fine-tuning functionality to optimise hyperparameters and data generation parameters.
- a comparison between simple orthographic projection and perspective projection to generate images for 2D deep learning classification framed on Gestalt approach.
- the use of surface curvature feature for image enrichment and the limitations of this feature in the point cloud domain.

The remainder of the paper is organized as follows: section 2 summarizes the previous related work. The proposed method is theoretically described in section 3. Then, results and experiments are shown and discussed in section 4. Finally, section 5 is devoted to conclude the work.

2. Related work

This section addresses the state of the art of point cloud deep learning classification focusing on techniques to tackle the labelling problem: Data Synthesis and Transfer Learning applied to 2D data.

High performance proved by CNNs in image classification together with the increased availability of point clouds have led researchers to the development of new 3D deep learning architectures. Nevertheless, handling the non-regular structure of point clouds is still a challenge for CNNs. Hence previous works have addressed this hardship with different strategies which can be classified in three categories according to how data is represented. *Multi-view* or projection-based approach consists in generating multiple 2D images from several perspectives of the point cloud with the aim to take advantage of the yield attained by 2D CNNs (Su et al., 2015). The main drawback of these methods is the loss of both spatial information of the objects and spatial relationship to each other due to the 2D projection. Within 3D approaches, *volumetric methods* (Maturana and Scherer, 2015; Tchapmi et al., 2017; Le and Duan, 2018) have been proposed to benefit from 3D geometrical object representation while preserving the nature of convolutional operations from 2D CNN. For this purpose, point clouds are previously converted to a 3D regular structure, commonly composed of voxels. Despite the more realistic geometric representation, classification using the multi-view CNNs (MVCNNs) has proved higher performance than volumetric CNNs (Ruizhongtai Qi et al., 2016) due to the lower resolution of voxel representation among other factors. An alternative to structured approaches consists in the use of network architectures capable of taking the points directly as input data (Charles et al., 2017; Qi et al., 2017). The performance of early architectures developed on the basis of this *unstructured approach* was lower than the structured approaches. However, in the last years, the improvement of theses architectures has been an active research topic are achieving promising results (Landrieu and Simonovsky, 2018).

The shortage of 3D labelled data for training these 3D networks is an important disadvantage compared to 2D CNNs (Griffiths and Boehm, 2019b). Point cloud object labelling is a time-consuming task since each point that belongs to the object needs to be pin pointed. Although some datasets composed of labelled point cloud data generated from RGB-D images (Silberman et al., 2012; Xiao et al., 2013; Armeni et al., 2017; Chang et al., 2017) or 3D CAD models (Wu et al., 2015) are available, the amount of classified data is far from the number of images provided by the popular dataset ImageNet (Krizhevsky et al., 2012) which is composed of over 14 million labelled images. To deal with 3D data labelling problem, this work proposes a combination of Data Synthesis and Transfer Learning techniques applied to 2D data.

In this line, data synthesis is gaining preference to solve the problem of

relevant and labeled data lacking. In fact, some areas of data mining and machine learning rely significantly on benchmark data sets to compare and evaluate results across competing methods in their development. To overcome the shortage of publicly available large real-world data sets, synthetic or semi-synthetic datasets are very suitable for a wide range of controlled scenario tests for several Machine Learning methods, such as regression (Cano and Torra, 2009), classification (Sánchez-Monedero et al., 2013), drift detection (Iglesias et al., 2020; Belford et al., 2017), anomaly detection (Taylor et al., 2016), failure prediction (Hajiaghayi and Vahedi, 2019). In fact, this approach brings the possibility of producing a large quantity of relevant data, of generating data with desired characteristics for testing and of knowing exactly the targets and the underlying models. In some cases, for specific complex data algorithms such as Multi-Label Classification and Multi-target Regression, the dataset may be difficult to find (Tomás et al., 2014). The main drawback of Data Synthesis is that it does not have a well-defined methodology since there is not a specific technique for each data type in each context. Besides, as this does not generate all the diversity of data and the true noise information, the required models precision may be not well represented.

But this approach goes a step further. Through data synthesis it is possible to teach the concept of a class to a training model. That is, data is generated in a way that expresses the main features of classes. In particular, in this work, furniture objects are characterized in their essential features, primarily their geometry, that we use to identify the same objects. This approach is inspired in the Gestalt theories of perception, where human perception tends to understand objects as an entire structure rather than the sum of its parts. Therefore, objects are synthesized by using overall shapes that include all parts of the object.

Transfer learning is also another technique to solve the problem of training data shortage. Particularly, 2D neural networks are less likely to overfitting than the 3D architectures due to the amount of available labelled images in contrast 3D models. To leverage the potential of 2D CNNs, the synthetically generated point clouds are transformed into images, so that this approach can be also a particular case of transfer learning. In fact, data does come from the domain of point clouds while training is performed with synthetic images. A closed transfer learning approach was used by Balado et al. (2020) proving that augmenting RGB image dataset generated from coloured point clouds with images avaliable on online resources can improve furniture object

classification. Goyal et al. (2021) used depth images generated from point clouds obtaining results in the state-of-the-art. Our method generate synthetic point clouds from BIM objects. Resultant point clouds are mapped to binary and greyscale images by applying orthographic and perspective projection. Results obtained using the different generated images are evaluated and compared.

3. Method

The general workflow of the proposed method is depicted in Fig. 1. The method starts by collecting BIM objects from websites and organizing them into object classes to serve as source dataset. After, these models are processed to generate a synthetic dataset of classified point clouds affected by different perturbations such as noise and occlusions. Once labelled point clouds are generated, they are used as the input data to the deep learning framework developed for point cloud object recognition. This framework is composed of sequential steps such as image generation, model computation and object classification which are explained in more detail in the following sections.

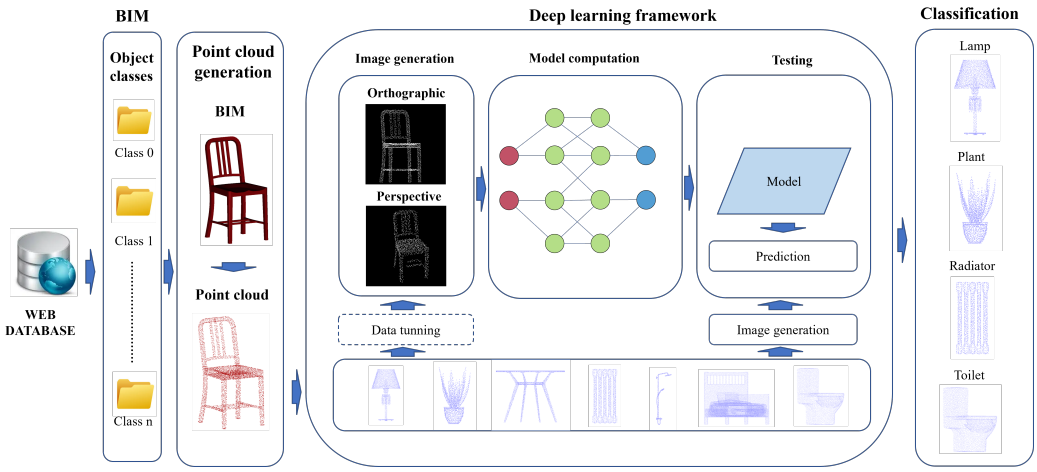

Figure 1: General workflow

3.1. Point cloud data generation

In this step, a classified point cloud dataset is generated by processing BIM objects which are downloaded from website databases. Unlike virtual

CAD models, object models provided by manufacturers correspond to real-world objects. Thus, these models are downloaded from free registration websites and stored in a directory arranged by object category. The use of graphical interfaces of database search engines facilitates model collecting process by type object filtering. Besides, file format is restricted to those that do not require commercial software to be processed.

Both BIM models and point clouds are handled and processed by the open-source python library Open3D (Zhou et al., 2018). This software provides the necessary transformations and operations for the method development. BIM models with OBJ. format are directly imported to be processed while *ifc* format models are automatically converted to OBJ. by invoking the open-source application *IfcConvert* in batch mode.

Then, from each object model, four point clouds are generated by adding perturbations with different characterisations. First, a point cloud without any perturbation is directly obtained by poisson disk sampling algorithm (Yuksel, 2015) which requires determining the number of sampled points n_{pts}. To ensure a fixed density for every object point cloud, n_{pts} is calculated regarding the total area of the object surface A_S by the following equation:

$$n_{pts} = dens * A_S \tag{1}$$

Where $dens$ is a fixed density in pts/m^2.

After, object point cloud orientation is visually checked and corrected if the object is not aligned with Z axis. This step is required to augment the data by applying coherent rotations along Z axis in the sense of how the objects are generally positioned in the real-world. This oriented point cloud P_o is an instance of the perturbed-free dataset which will be used to generate training/validation images. Next, three perturbed datasets are generated from P_o by adding noise and occlusions to P_o. Synthetic data simulates these real effects presented in point clouds serving as useful training data to classify real data or as testing data when real data is not available.

Fig. 2 depicts how the sampled points clouds are processed to generate the four characterised datasets. Noisy point clouds are derived by Gaussian noise addition operation which consists in aggregating an offset to each point position belonging to a real surface. The added noise is modeled by a zero-average normal distribution with a standard deviation σ.

The presence of occlusions in point clouds is a common hardship in real scenarios due to the appearance of objects between the laser beam and the

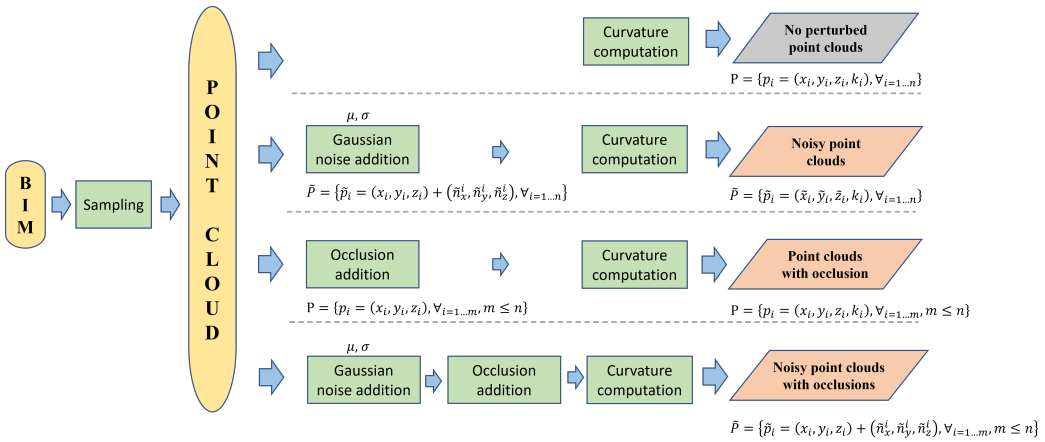

Figure 2: Generation of four types of point clouds according to the perturbations implemented.

target object. To take into account this perturbation, the process of adding occlusion intends to simulate occlusion effects by using a fast and simple visibility analysis known as Hidden Point Removal (HPR) which is implemented in the Open3D library. As shown in Fig. 3, four subsets of points are extracted from the input point cloud P_i on basis of visibility from four viewpoints. The bounding box enclosing the object BB_{obj} is taken as a reference to determinate the observation positions. These view points belong to a plane located d_{xy} meters away from vertical faces of the BB_{obj} and orthogonal to the faces. To determinate view positions, centroids of vertical faces are projected to the planes and then projected points are vertically shifted by arbitrarily setting point height to Z_{off} relative to the bottom of the BB_{obj}. Finally, occlusion is simulated by discarding a point subset randomly selected so that the resulting occluded point cloud is composed of the three remaining point subsets.

The last dataset consists of point clouds characterised by both noise and occlusion. Thus, this data are the most challenging for object recognition as shown in Fig. 4 illustrating the processes implemented to generate four types of point clouds from the same object model.

Once synthetic datasets are created, 3D information provided by point clouds is exploited to extract object features that can increase classification performance. Considering that the approach of this work is based on Gestalt principle, variation of point cloud surfaces can provide useful information to

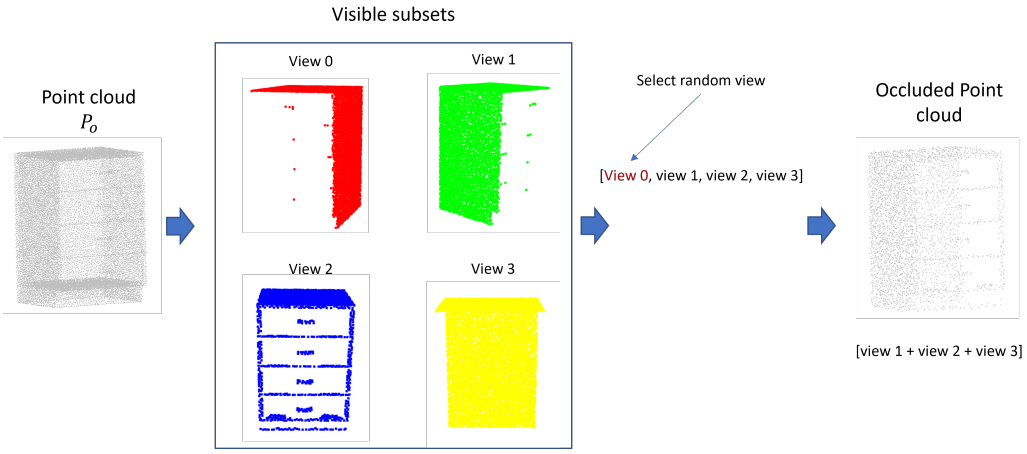

Figure 3: Occlusion addition process.

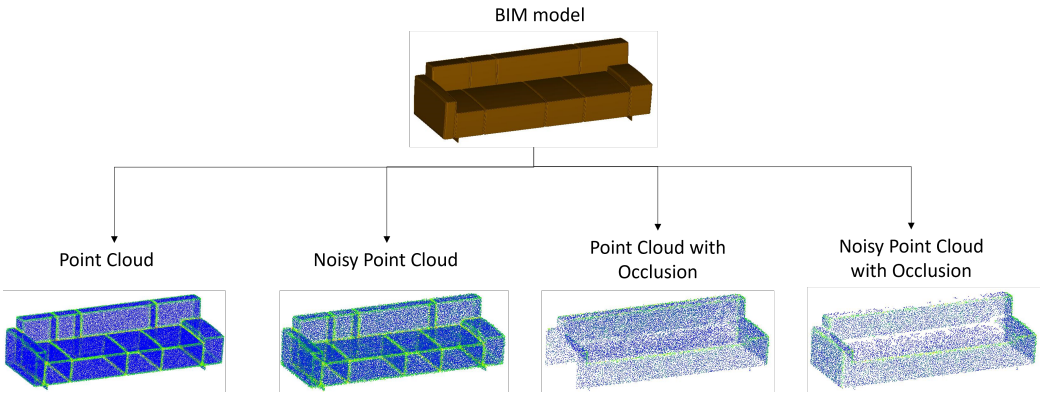

Figure 4: Example of the four point clouds obtained with different characterisations from a BIM model object

define the whole object. Therefore, surface variation is computed from each point by applying the method proposed by Bazazian et al. (2015) based on the eigenvalues of the covariance matrix. Unlike original method, radius search criteria is used to select neighboring points which are contained in a closed ball of radius ρ centered on the evaluated point. Point clouds visualised in Fig. 4 are coloured according the computed surface variance value.

In summary, point cloud generation process converts 3D BIM objects into different characterised point clouds providing synthetic classified datasets. The process is automated requiring human-interaction only to collect models

from websites and to carry out visual inspection to discard bad-built models or to orient points clouds if necessary. Particularly, in this work, the generated synthetic point cloud dataset is used for training a neural network and testing the trained model input data.

3.2. Deep learning framework

A flexible deep learning framework has been implemented with the aim of conducting experiments using differently characterised point clouds, several neural network architectures and optimisation techniques for hyperparameter or data tuning. First, input object point clouds are converted to suitable data for the neural networks integrated on the framework that work with 2D images. Previously, image generation parameters can be optimised if real data are provided. Once training, validation and testing images are created, a deep learning model is used for object prediction. The model can be computed by training a neural network from scratch with training/validation images or with auto Machine Learning to fine-tune network hyperparameters. Also, a pre-trained model can be loaded to streamline tests within the similar domain. Fig. 5 depicts deep learning framework procedures which are exposed more extensively below.

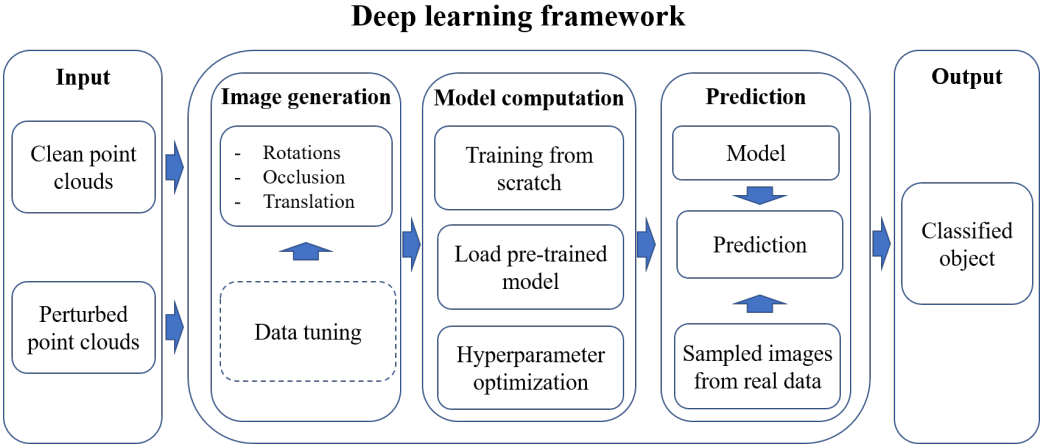

Figure 5: Workflow of the deep learning framework

3.2.1. Image data generation

The initial process on the framework consists in transforming point clouds to 2D image format adopted by the embedded deep learning architectures. A

commonly used technique to represent three-dimensional data into 2D is 3D graphical projection encompassing various transformations to generate images. The framework incorporates two selectable projections: orthographic and perspective. While images generated by orthographic projection, hereinafter referred as 'orthographic images', are derived by directly projecting points to XZ plane, perspective projection requires fixing a camera optical center and determining a projection plane.

Images generated by perspective projection, which will referred to as 'perspective images' are more realistic from the human-perception of the object than orthographic images (Hearn and Baker, 1997). The camera optical center O_c is fixed to a horizontal distance d_h from a vertical face of BB_{obj} at Z_{off} meters height from the bottom of BB_{obj}. Then, projection plane Π_{proj} is defined by the normal vector from object center C_o to O_c and the bottom vertices of the vertical face. Next, visible points from O_c are determined by HPR algorithm and they are projected on the Π_{proj} to derive a 2D object representation.

Fig 6 visualises the workflow of the entire image generation process from non perturbed point clouds to multi-channel images. In order to take advantage of the fact that the source data are represented on the 3D space, several 2D images can be generated from each instance by making a random rotation along the object up vector. From this randomly rotated point cloud, n_{rot} arbitrary rotations are carried out to generate one projected image for each rotation corresponding to image channels. Before projection, noise and occlusion addition processes described in 3.1 are applied on rotated point clouds if these are perturbed free and testing data are real. For perspective projection, occlusion addition is omitted because this perturbation is intrinsic to the visible points calculation.

After projection, points can be mapped into both binary or greyscale pixels depending on whether point cloud features are regarded or not. Pixels composing binary images take value 1 if at least one projected point falls in the pixel area and the value 0 is assigned otherwise. In greyscale images, pixels represent the above computed surface variation of points projected on the pixel. Previously, values of surface variation are scaled using Eq. 2 to avoid that points with no change in the surface (value 0) are represented as empty pixels on the greyscale image:

$$\sigma_r^{'}(p) = \sigma_r(p) + (1 - \max\{\sigma_r(p_i) : p_i \in P, i = 0, ..., n-1\}) \qquad (2)$$

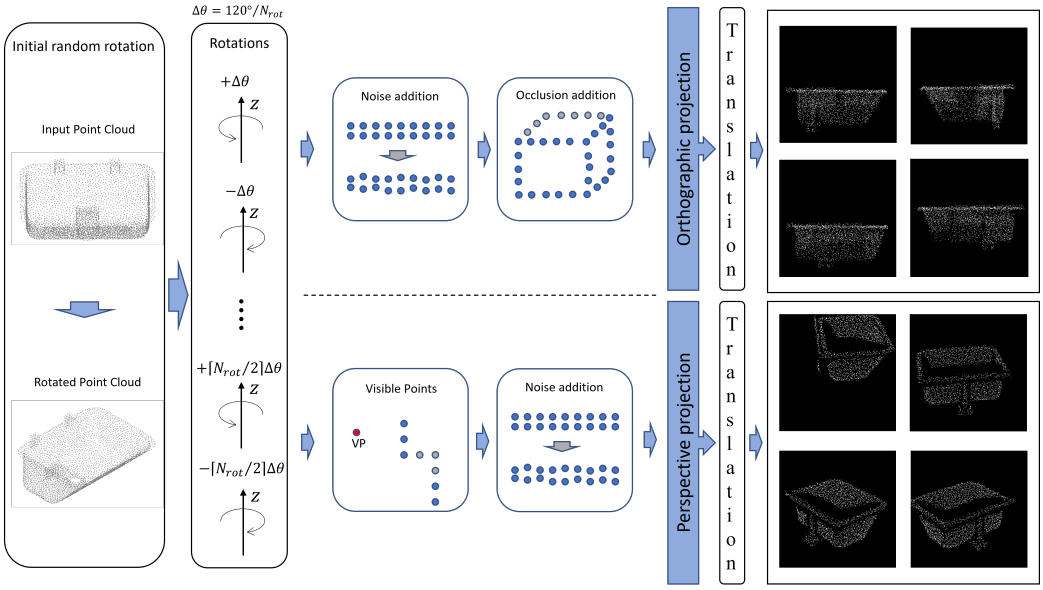

Figure 6: Workflow of image generation process.

Where $\sigma_r(p_i)$ represents the surface variation on the point p_i considering a neighbourhood radius $/r$. The pixel value is the averaged scaled surface variation of points projected on the pixel. Lastly, pixels are randomly translated along to horizontal and vertical axis at most T_{h_max} and T_{v_max}.

Examples of both image formats depicted in Fig 7 show that surface changes along the object are more highlighted in greyscale images.

The number of images generated for training n_{train}, validation n_{val} and testing $n_{testing}$ are arbitrarily set being independent of the size of the point cloud dataset. Previously to image generation, parameters used for image generation such as maximum rotation angle Θ_{max}, σ, T_{h_max}, T_{v_max} can be obtained from real by data tuning process which is further explained in the next section.

3.2.2. Model computation

At the core of the developed approach, both synthetic data generation and model training are presented. As will become clear in the following paragraphs, both are deeply connected by a parameter optimisation methodology, which sweeps across most of the developed modeling work.

Given the extreme computational costs of generating large enough datasets for deep learning and of creating and training deep learning models, an opti-

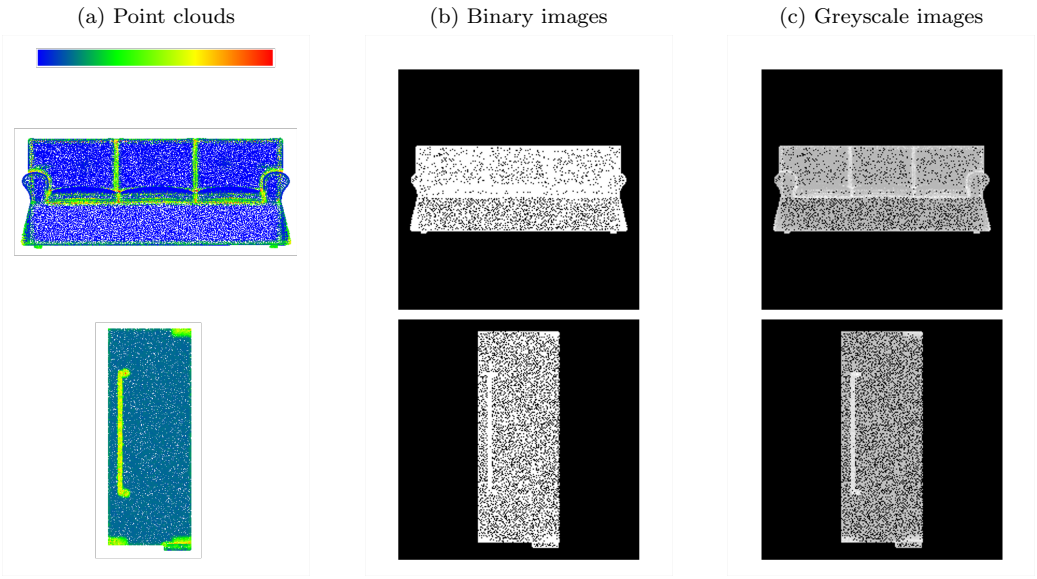

Figure 7: Representation of two objects as a) point cloud coloured on basis its surface variation, b) binary image and c) greyscale image.

misation method which aims at reducing the number of parameter configurations tested or that is able to intelligently decrease the cost of these different operations is a mandatory requirement.

From the large amount of optimisation and auto ML methodologies one in particular was found to be very promising, not only presenting the best results of all research gathered, but also providing the full source code, thus giving a very strong starting position.

This optimisation method is Harmonica by (Hazan et al., 2017). We will provide a description of how this method operates, however a technical and detailed explanation is outside the scope of this paper and thus the interested reader should refer to the aforementioned work.

The basic premise of Harmonica is to identify the most important parameters and to lock their values to the ones which lead to the best performance. Then the optimisation of the remaining parameters follows in this now reduced search space. It works on the following manner:

Step 0 Given a set of tunable parameters $X = (x_1, x_2, ..., x_n)$ with each parameter $x_m \in \{0, 1\}$ and a cost function $f(X)$.

Step 1 Randomly initialize X and compute the score $= f(X)$.

Step 2 Repeat step 1 a fixed N number of times, storing all parameters sets and scores.

Step 3 Expand the parameter sets with polynomial combinations including interactions. Run Lasso regression with the list of polynomials as the feature and the list of scores as the target.

Step 4 Use Lasso coefficients to identify monomial importance (powers and interactions of features). Extract only the top n monomials.

Step 5 Obtain the set of important features. Important features are those contained in the monomials obtained in step 4.

Step 6 Obtain all the possible value combinations for the set of important features.

Step 7 For each combination obtain the score. The score for a value combination C_i is given by $\sum_{j=1}^{R}\{Coef_j \times \prod_{k=1}^{S}\{v_k : v_k \in C_i, f_k \in M_j\}\}$, with R and S being the number of monomials and important features, M and Coef being the monomials and respective coeficients, f being an important feature and v being the value of the combination.

Thus, the score of the combination is the sum of the scores of the monomials. While the score of a monomial is the coeficient of the monomial times the multiplication of the values for each feature in the monomial.

Step 8 Extract the top n combinations by score.

Step 9 Repeat steps 1-9 K times, saving all the combinations obtained in step 8.

Step 10 Delegate further optimisation to a different optimisation method, passing the obtained value combinations that should be locked.

Step 11 Return the best set of parameters found.

Given this it is possible to see that despite the great performance, the functioning of the method is quite intuitive. The attentive reader might however have noticed that there are two components that have yet to be fully defined.

The first one is the cost function, which being dependent on the exact problem and application will be different for the data parameter tuning and model hyperparameter optimisation steps. Both these cases will be detailed further into this section.

The second component is the optimisation function to which the remaining parameter optimisation is delegated. Like the cost function this too can be changed, needing only to modify any intended function to lock the passed value combinations in place. In the work where Harmonica is presented two alternatives are proposed. The first one is simple random search, with the non locked parameters being randomly set. This is, of course, not a very intelligent strategy, which nevertheless tends to obtain quite good results. The second alternative, and the one which we opted to use in this work, is Hyperband, presented by (Li et al., 2017).

Much like for Harmonica we will present a brief overview of the core idea behind the Hyperband method and an explanation of its functioning, however for more technical details we direct the reader either to the harmonica paper, or to the far more detailed Hyperband paper referred before.

The basic premise of Hyperband is first to reduce the amount of resources allocated to parameter configurations which are not promising, by stopping these early, while letting those which quickly yield better results continue. This can be done by for example only letting models train for a small number of iterations, and continue training only for the ones obtaining the best results, or by allocating only limited time to a operation.

However, different problems converge at different speeds, so for some a partial resource allocation might be sufficient to differentiate, while for others it might not. Thus, the second part of Hyperband tackles this problem by testing different allocations and pruning speeds, from simply allocating all resources equally to all candidates, like random search, or on the other extreme allocating the minimum resources to many candidates and only continuing with one. As such, it works in the following manner:

Step 0 Given a total budget B, a maximum budget per candidate Bc, a control parameter η and, for the specific case of using Harmonica, the list of locked configurations Lc.

Step 1 Calculate the number of pruning strategies to test $S = \left\lfloor \frac{\log Bc}{\log \eta} \right\rfloor$ and the budget per strategy $b = \left\lfloor \frac{B}{S+1} \right\rfloor$.

Step 2 For all values of $s \in \{n : n \in \mathbb{N}, 0 \leq n \leq S\}$, perform steps 3-5, saving the best parameters.

Step 3 Calculate the number of initial random configurations $Cn = \left\lfloor \frac{B \times \eta^s}{Bc \times (s+1)} \right\rfloor$. Obtain the initial budget for each configuration $b_0 = \lfloor Bc \times \eta^{-s} \rfloor$.

Step 4 For all values of $i \in \{n : n \in \mathbb{N}, 0 \leq n \leq s\}$, perform step 5, saving the best parameters.

Step 5 Create Cn random configurations, picking, if using Harmonica, one element of Lc at random for each. Obtain the cost for all configurations.

Step 6 With R being the number of remaining configurations. Keep only $\left\lceil \frac{R}{\eta} \right\rceil$ of the best configurations. Update the budget per configuration for the next iteration to $b_i = \lceil b_{i-1} \times \eta^s \rceil$.

Step 7 Return the best parameters found.

Similarly to Harmonica, the Hyperband algorithm achieves great results while being quite intuitive, simply stopping less promising configurations early, allowing better resource allocation to all others. Given this, we will now present the two instances where this method was employed, starting with data generation parameter optimisation.

As the basic version of Harmonica can only handle binary parameters, to define more detailed parameter grids multiple binary parameters must be combined. A total of 11 parameters, mapped into 4 actual parameters were employed. These are 4 for noise intensity, 3 for maximum random z-rotation, and 4 for maximum translation, with 2 for each axis (horizontal and vertical).

The cost function thus works as follows. First, a random instanciation of the binary parameters is obtained by the optimisation method, with these then being mapped into the actual parameters. The synthetic dataset is generated with these parameters and a predefined deep learning model is trained to classify the instances of this dataset. Finally, the validation performance in terms of accuracy is obtained for the classification of the actual, real data.

For the hyperparamter optimisation of the deep learning models the process is quite similar. A synthetic dataset, that can be either be parameter optimised or manually defined, is created. Then, the binary parameters are set by Harmonica. There are a total of 8 parameters, 3 for learning rate, 2 for batch size, 1 for momentum existence, 1 for momentum value and 1 for the selection of the optimisation method. With these, the values are mapped and the model trained on the synthetic dataset. Finally, the model is validated on a different partition of the synthetic dataset and the accuracy is returned.

For this task, a set of deep learning models which have proven to be very effective in computer vision were selected. All the models are presented in

table 1[1], which span from the very simple 8 layer LeNet to the 43 million parameter Inception-v4. Unlike for the various parameters, model selection was done manually and separately, with architectures showing less promise, like VGG-16 and Inception-v3 having less time dedicated to tuning.

To wrap up this section, we would be remiss to not point out a concerning detail that the discerning reader might have noticed. This is the fact that during the dataset parameter tuning, real data is used as a validation set, in order to obtain the final score. Due to the data driven optimisation methodology this was required, but could, if mishandled, lead to a form of parameter overfitting. To combat this two constraints were put in place. First, there is no intersection between the portion of the real data used in the optimisation and the one used for the final results. And second, the model architecture used for data parameter tuning is always different from the one optimised and used for the final predictions, with the former usually being the simpler LeNet or AlexNet, and the latter being one of the other 6.

Table 1: Description of the neural networks available on the framework

Architecture	**Description**
LeNet (LeCun et al., 1989)	Simple convolutional network for handwritten digits.
AlexNet Krizhevsky et al. (2012)	Winner of ILSVRC-2012 in image classification and localization.
VGG-16 (Simonyan and Zisserman, 2014)	Archive the first and second place of ILSVRC-2014 in image classification and localisation.
Inception-v1 (Szegedy et al., 2015)	Presents a novel block-based architecture with important performance improvments.
Inception-v3 (Szegedy et al., 2016)	Runner-up in ILSVRC-2015.
Inception-v4 (Szegedy et al., 2017)	Major performance and computational cost improvemnts over previous versions.
resNet 50 (He et al., 2016)	One of the first to include skip connections, greatly increase potential performance.
Xception (Chollet, 2017)	The first major network composed only of depthwise separable convolution layers.

4. Results

This section is organized in two parts in accordance with the workflow of the method. First, the results of data generation are presented. The second describes the tests performed and the results are discussed.

4.1. Dataset

The generated dataset is composed of 21 object classes collected in table 2. The number of instances in each class is less than the downloaded objects since bad-built and mis-scaled models have been discarded during point cloud

[1]The article by (Karim, 2020) contains a quick, yet comprehensive, description of the used architectures, and was partially responsible for the selection made.

generation. Furthermore, because classification is performed based on the geometry of the object, the classes were defined taking into account geometrical features. Thus, chandeliers are included in a specific class instead of in the general lamp class because chandelier geometry is much more complex than common lamps. On the contrary, two objects of different type with similar geometry such as washing machines and dryers are considering in the same class.

Table 2: Information about classes composing the dataset.

Class	**Nº instances**	**Class**	**Nº instances**	**Class**	**Nº instances**
Bed	20	Door	28	Shower	78
Bidet	79	Drawer	25	Sink	73
Bin	31	Fridge	11	Sofa	76
Bookcase	4	Fume extractor	9	Table	89
Chair	71	Lamp	50	Toilet	137
Chandelier	19	Plant	20	Urinal	70
Chimney	10	Radiator	26	Washing machine/Dryer	12

Examples of point clouds obtained by point generation process are depicted in Fig. 8 where points are coloured on the basis of point surface variation. The fixed density *dens* used to sample point clouds was $4000pt/m^2$. Concerning perturbation addition, Gaussian noise was modeled by a normal distribution with a standard deviation of 0.005m while the parameters d_{xy} and z_{off} required to determinate the viewpoints in occlusion addition process were set to 2.0 m and 1.5 m respectively. For surface variation computation, the radius r was set to 0.04 m.

4.2. Tests and classification results

From the entire dataset described in the above section, four different reduced datasets consisting of 5, 6, 8 and 10 object classes were created. Table 3 lists the classes selected for each dataset. The point clouds selected for each class are composed of 80% non-perturbed instances which are used to training/validation and 20% noisy point clouds with occlusion simulating real data in the testing stage.

For deep learning framework configuration, AlexNet architecture was selected for training data setting epochs and batch size to 18 and 128 respectively in every experiment. For image generation, four rotations have been carried out for each image considering a maximum rotation angle Θ_{max} of

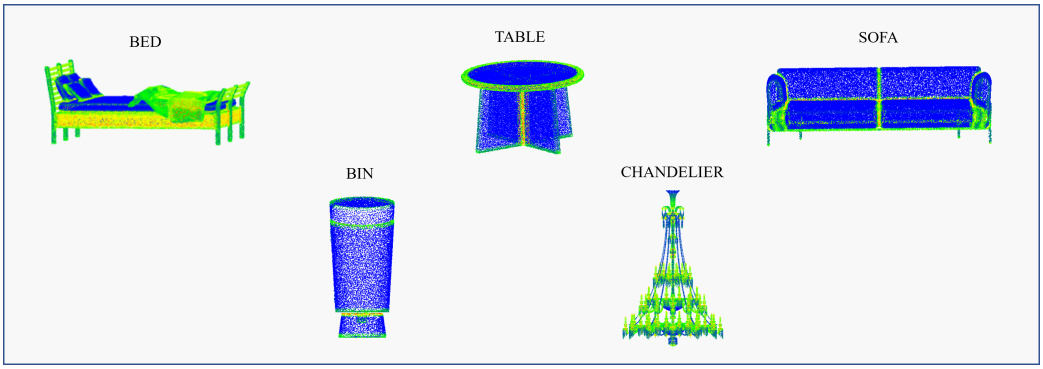

Figure 8: Examples of generated point cloud coloured on the basis of point surface variation.

Table 3: Class lists composing datasets.

Dataset	Classes
Dataset1	chandelier, chimney, fridge, shower, toilet
Dataset2	bin, shower, sink, sofa, table, urinal
Dataset3	bin, chair, chandelier, door, shower, sink, table, toilet
Dataset4	bed, bidet, chair, chimney, fridge, radiator, shower, plant, sink, table

120^{o}. Resolution of both binary and greyscale images was 224x224 while the number of images generated for training/validation was 1500/400, 1500/400, 2000/600, 2400/1000 for Dataset1, Dataset2, Dataset3 and Dataset4 respectively.

For each dataset, four simulations have been carried out to assess how 3D projection and format image influence on the trained model performance. Table 4 lists confusion matrices derived from classification of Dataset1 using orthographic and perspective projection to generate binary images. Classification performance was measured by overall accuracy (OA) metric which is the ratio between the number of objects correctly classified and the total objects. Results show that the neural network recognizes better the object using orthographic images than perspective images.

Similar tests were carried out generating greyscale images on basis of surface variation feature. Confusion matrices collected in Table 5 show that the OA increases with both orthographic and images.

The same 4 tests were replied with the remaining datasets. The OA obtained in each test is represented in Fig 9. With regard to the graphical

Table 4: Confusion matrices of object prediction in dataset1

	ORTHOGRAPHIC					**PERSPECTIVE**				
Class	Cha	Chi	F	S	T	Cha	Chi	F	S	T
Chandelier(Cha)	2	-	-	-	-	2	1	-	-	3
Chimney(Chi)	-	2	-	-	-	-	1	-	-	-
Fridge(F)	-	-	2	-	-	-	-	1	-	-
Shower(S)	1	-	-	15	-	1	-	-	15	-
Toilet(T)	-	-	-	-	27	-	-	1	-	24
Accuracy	**0.98**					**0.88**				

Table 5: Confusion matrices of object prediction in dataset1 using greyscale images

	ORTHOGRAPHIC + SV					**PERSPECTIVE + SV**				
Class	Cha	Chi	F	S	T	Cha	Chi	F	S	T
Chandelier(Cha)	3	-	-	-	-	3	-	-	-	2
Chimney(Chi)	-	2	-	-	-	-	2	1	-	-
Fridge(F)	-	-	2	-	-	-	-	1	-	-
Shower(S)	-	-	-	15	-	-	-	-	15	-
Toilet(T)	-	-	-	-	27	-	-	-	-	25
Accuracy	**1.0**					**0.94**				

projection used to generate the images, all tests prove that orthographic projection (blue line) is more suitable than perspective projection (green line) for point cloud object classification based on Gestalt approach. This find is opposite to the expected result, since perspective projection provides more realistic images being more distinguishable for the human-eye. The reason for this counter-intuitive outcome may be due to the simpler object representation in orthographic images making the whole form of the objects more clearly perceptible.

The use of greyscale images based on surface variation instead of binary images increases the classification performance in most cases. However, this feature might be not useful for classification in some cases on account of limitations on surface variation computation. Perturbations added to point cloud such as noise and occlusion may affect to surface variation computation as shown in Fig. 10a. Furthermore, the robustness of surface variation

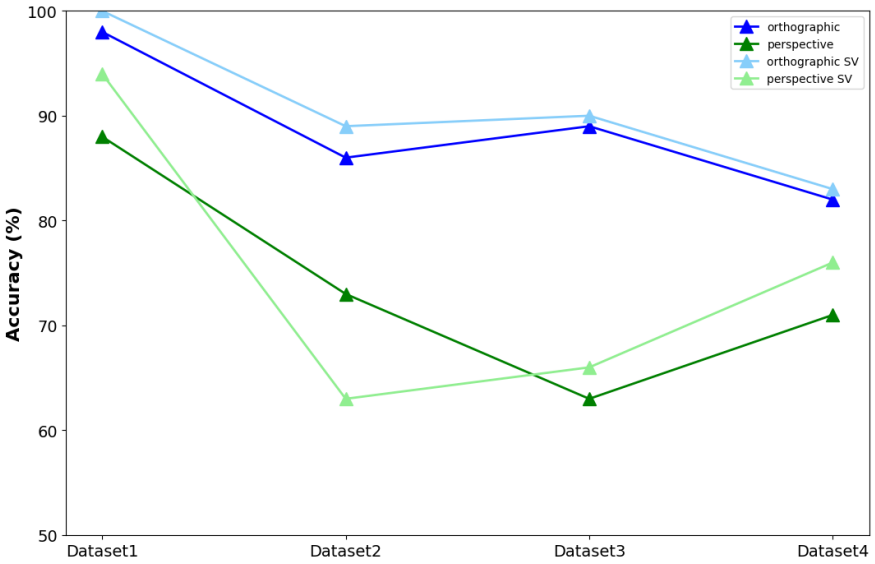

Figure 9: Representation of the overall accuracy obtained for each test using orthographic binary images (blue line), orthographic greycale images (light blue), perspective binary images (green) and perspective greyscale images(light green).

algorithm decreases for objects with smooth variation surfaces (Fig 10b).

Another shortcoming is related to the HPR algorithm used for occlusion addition and the determination of visible points in perspective image generation. Since these operations have to be run multiple times to prepare data training/validation in execution time, HPR algorithm is suitable for the implemented method because it is fast and simple. However, high level of noise and curvature negatively affects the correct determination of whether a point is visible or not.

5. Conclusions

A highly automated method for generating synthetic classified point clouds from BIM objects has been presented. The synthetic point cloud datasets are characterised by various simulating perturbations present in real data such as noise or occlusions. Furthermore, a deep learning framework was implemented to evaluate the feasibility of using this synthetic data for object

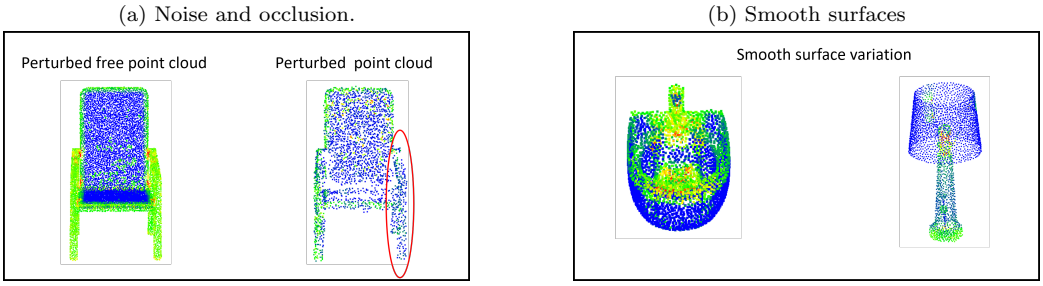

Figure 10: Effect of a) noise, occlusion and b) smooth surfaces in surface variation computation.

classification with deep learning techniques. Data generated in this way are significantly helpful for augmenting dataset, using data controlled or balancing datasets by adding instances of shorter classes.

Results revealed the OA of deep learning models is higher when model is trained with orthographic images instead perspective images. Beyond projection, the use of surface variation feature to enrichment greyscale images when RGB data is not provided was explored. This object representation improved classification performance in almost all tests. However, surface computation limitations on complex point clouds might generate a misleading point cloud mapping. In future work, the study of new methods using different data sources should be addressed with the aim to develop a general methodology that produces reliable synthetic point cloud for deep learning classification. In addition, attributes such as contextual information could be added to overcome geometric representation weaknesses. Particularly, the proposed method could be extended to generate point cloud scenes to use in point cloud segmentation.

6. Acknowledgments

This work has been partially supported by the European Association of International Cooperation Galicia-North of Portugal, program IACOBUS VI through a stay at INESC TEC to E. Frías (IACOBUS VII-35563149). This project has received funding from the Xunta de Galicia through project ED431C 2020/01, and from the Government of Spain through project PID2019-105221RB-C43 funded by MCIN/AEI/10.13039/501100011033 and through human resources grant RYC2020-029193-I funded by MCIN/AEI/10.13039/501100011033 y FSE "El FSE invierte en tu futuro". The open access fee has

received funding from the University of Vigo/CISUG. This document reflects only the views of the authors. The statements made herein are solely the responsibility of the authors.

References

Abanda, F., Byers, L., 2016. An investigation of the impact of building orientation on energy consumption in a domestic building using emerging bim (building information modelling). Energy 97, 517 – 527. doi:https://doi.org/10.1016/j.energy.2015.12.135.

Abdel-Hamid, O., Mohamed, A., Jiang, H., Penn, G., 2012. Applying convolutional neural networks concepts to hybrid nn-hmm model for speech recognition, in: 2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pp. 4277–4280. doi:10.1109/ICASSP.2012.6288864.

Alawadhi, M., Yan, W., 2021. BIM Hyperreality: Data Synthesis Using BIM and Hyperrealistic Rendering for Deep Learning. arXiv e-prints , arXiv:2105.04103arXiv:2105.04103.

Armeni, I., Sax, A., Zamir, A.R., Savarese, S., 2017. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. ArXiv e-prints arXiv:1702.01105.

Balado, J., Díaz-Vilariño, L., Verbree, E., Arias, P., 2020. Transfer learning for indoor object classification: From images to point clouds. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences 5, 65–70. doi:10.5194/isprs-Annals-V-4-2020-65-2020. 2020 24th ISPRS Congress - Technical Commission IV on Spatial Information Science ; Conference date: 31-08-2020 Through 02-09-2020.

Bazazian, D., Casas, J., Ruiz-Hidalgo, J., 2015. Fast and robust edge extraction in unorganized point clouds, pp. 1–8. doi:10.1109/DICTA.2015.7371262.

Belford, M., Namee, B.M., Greene, D., 2017. Synthetic dataset generation for online topic modeling, in: CEUR Workshop Proceedings.

Bosché, F., Guillemet, A., Turkan, Y., Haas, C., Haas, R., 2013. Tracking the built status of mep works: Assessing the value of a

scan-vs.-bim system. Journal of Computing in Civil Engineering 28. doi:10.1061/(ASCE)CP.1943-5487.0000343.

Bowyer, K.W., Chawla, N.V., Hall, L.O., Kegelmeyer, W.P., 2011. SMOTE: synthetic minority over-sampling technique. CoRR abs/1106.1813. arXiv:1106.1813.

Broquetas, M., Bryde, D., Volm, J., 2013. The project benefits of building information modelling (bim). International Journal of Project Management 31, 971–980. doi:10.1016/j.ijproman.2012.12.001.

Cano, I., Torra, V., 2009. Generation of synthetic data by means of fuzzy c-regression, in: IEEE International Conference on Fuzzy Systems. doi:10.1109/FUZZY.2009.5277074.

Chang, A., Dai, A., Funkhouser, T., Halber, M., Niebner, M., Savva, M., Song, S., Zeng, A., Zhang, Y., 2017. Matterport3d: Learning from rgb-d data in indoor environments, in: 2017 International Conference on 3D Vision (3DV), pp. 667–676. doi:10.1109/3DV.2017.00081.

Charles, R.Q., Su, H., Kaichun, M., Guibas, L.J., 2017. Pointnet: Deep learning on point sets for 3d classification and segmentation, in: 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 77–85. doi:10.1109/CVPR.2017.16.

Chollet, F., 2017. Xception: Deep learning with depthwise separable convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1251–1258.

Dey, A., 2016. Machine learning algorithms : A review.

Goyal, A., Law, H., Liu, B., Newell, A., Deng, J., 2021. Revisiting point cloud shape classification with a simple and effective baseline. arXiv:2106.05304.

Griffiths, D., Boehm, J., 2019a. A review on deep learning techniques for 3d sensed data classification. Remote Sensing 11, 1499. doi:10.3390/rs11121499.

Griffiths, D., Boehm, J., 2019b. A review on deep learning techniques for 3d sensed data classification. Remote Sensing 11. doi:10.3390/rs11121499.

Hajiaghayi, M., Vahedi, E., 2019. Code failure prediction and pattern extraction using lstm networks, in: 2019 IEEE Fifth International Conference on Big Data Computing Service and Applications (BigDataService), pp. 55–62. doi:10.1109/BigDataService.2019.00014.

Hazan, E., Klivans, A., Yuan, Y., 2017. Hyperparameter optimization: A spectral approach. arXiv preprint arXiv:1706.00764 .

He, K., Zhang, X., Ren, S., Sun, J., 2016. Deep residual learning for image recognition, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770–778.

Hearn, D., Baker, M., 1997. Computer Graphics, C Version. Pearson Education.

Iglesias, F., Ojdanic, D., Hartl, A., Zseby, T., 2020. MDCStream: Stream Data Generator for Testing Analysis Algorithms, in: ACM International Conference Proceeding Series. doi:10.1145/3388831.3388832.

Jaritz, M., Gu, J., Su, H., 2019. Multi-view pointnet for 3d scene understanding, in: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops.

Karim, R., 2020. Illustrated: 10 cnn architectures.

Khoshelham, K., Díaz-Vilariño, L., Peter, M., Kang, Z., Acharya, D., 2017. The isprs benchmark on indoor modelling, in: Li, D. (Ed.), Proceedings ISPRS geospatial week 2017, 18-22 September 2017, Wuhan, China, International Society for Photogrammetry and Remote Sensing (ISPRS). pp. 367–372. doi:10.5194/isprs-archives-XLII-2-W7-367-2017.

Krizhevsky, A., Sutskever, I., Hinton, G., 2012. Imagenet classification with deep convolutional neural networks. Neural Information Processing Systems 25. doi:10.1145/3065386.

Landrieu, L., Simonovsky, M., 2018. Large-scale point cloud semantic segmentation with superpoint graphs. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition , 4558–4567.

Le, T., Duan, Y., 2018. Pointgrid: A deep network for 3d shape understanding, in: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9204–9214. doi:10.1109/CVPR.2018.00959.

LeCun, Y., Bengio, Y., Hinton, G., 2015. Deep learning. Nature 521, 436–44. doi:10.1038/nature14539.

LeCun, Y., Boser, B., Denker, J.S., Henderson, D., Howard, R.E., Hubbard, W., Jackel, L.D., 1989. Backpropagation applied to handwritten zip code recognition. Neural Computation 1, 541–551. doi:10.1162/neco.1989.1.4.541.

Li, L., Jamieson, K., DeSalvo, G., Rostamizadeh, A., Talwalkar, A., 2017. Hyperband: A novel bandit-based approach to hyperparameter optimization. The Journal of Machine Learning Research 18, 6765–6816.

Liu, W., Wang, Z., Liu, X., Zeng, N., Liu, Y., Alsaadi, F., 2016. A survey of deep neural network architectures and their applications. Neurocomputing 234. doi:10.1016/j.neucom.2016.12.038.

Ma, J.W., Czerniawski, T., Leite, F., 2020. Semantic segmentation of point clouds of building interiors with deep learning: Augmenting training datasets with synthetic bim-based point clouds. Automation in Construction 113, 103144. doi:https://doi.org/10.1016/j.autcon.2020.103144.

Maturana, D., Scherer, S., 2015. Voxnet: A 3d convolutional neural network for real-time object recognition, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), pp. 922–928. doi:10.1109/IROS.2015.7353481.

Pan, S.J., Yang, Q., 2010. A survey on transfer learning. IEEE Transactions on Knowledge and Data Engineering 22, 1345–1359. doi:10.1109/TKDE.2009.191.

Park, Y., Guldmann, J.M., 2019. Creating 3d city models with building footprints and lidar point cloud classification: A machine learning approach. Computers Environment and Urban Systems 75, 76–89. doi:10.1016/j.compenvurbsys.2019.01.004.

Perez, L., Wang, J., 2017. The effectiveness of data augmentation in image classification using deep learning. ArXiv abs/1712.04621.

Qi, C.R., Yi, L., Su, H., Guibas, L.J., 2017. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. CoRR abs/1706.02413. arXiv:1706.02413.

Ruizhongtai Qi, C., Su, H., NieBner, M., Dai, A., Yan, M., Guibas, L., 2016. Volumetric and multi-view cnns for object classification on 3d data, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 5648–5656. doi:10.1109/CVPR.2016.609.

Sánchez-Monedero, J., Gutiérrez, P.A., Pérez-Ortiz, M., Hervás-Martínez, C., 2013. An n-spheres based synthetic data generator for supervised classification, in: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). doi:10.1007/978-3-642-38679-4_62.

Silberman, N., Hoiem, D., Kohli, P., Fergus, R., 2012. Indoor segmentation and support inference from rgbd images, in: Fitzgibbon, A., Lazebnik, S., Perona, P., Sato, Y., Schmid, C. (Eds.), Computer Vision – ECCV 2012, Springer Berlin Heidelberg, Berlin, Heidelberg. pp. 746–760.

Simonyan, K., Zisserman, A., 2014. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 .

Su, H., Maji, S., Kalogerakis, E., Learned-Miller, E., 2015. Multi-view convolutional neural networks for 3d shape recognition doi:10.1109/ICCV.2015.114.

Szegedy, C., Ioffe, S., Vanhoucke, V., Alemi, A.A., 2017. Inception-v4, inception-resnet and the impact of residual connections on learning, in: Thirty-first AAAI conference on artificial intelligence.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2015. Going deeper with convolutions, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1–9.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S.E., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A., 2014. Going deeper with convolutions. CoRR abs/1409.4842. `arXiv:1409.4842`.

Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., Wojna, Z., 2016. Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2818–2826.

Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C., 2018. A survey on deep transfer learning. CoRR abs/1808.01974. arXiv:1808.01974.

Taylor, A., Leblanc, S., Japkowicz, N., 2016. Anomaly detection in automobile control network data with long short-term memory networks, in: 2016 IEEE International Conference on Data Science and Advanced Analytics (DSAA), pp. 130–139. doi:10.1109/DSAA.2016.20.

Tchapmi, L., Choy, C., Armeni, I., Gwak, J., Savarese, S., 2017. Segcloud: Semantic segmentation of 3d point clouds, in: 2017 International Conference on 3D Vision (3DV), pp. 537–547. doi:10.1109/3DV.2017.00067.

Tomás, J.T., Spolaôr, N., Cherman, E.A., Monard, M.C., 2014. A framework to generate synthetic multi-label datasets. Electronic Notes in Theoretical Computer Science doi:10.1016/j.entcs.2014.01.025.

Uy, M.A., Pham, Q.H., Hua, B.S., Nguyen, D.T., Yeung, S.K., 2019. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. arXiv:1908.04616.

Wang, J., Sun, W., Shou, W., Wang, X., Wu, C., Chong, H.Y., Liu, Y., Sun, C., 2014. Integrating bim and lidar for real-time construction quality control. Journal of Intelligent & Robotic Systems 79, 1–16. doi:10.1007/s10846-014-0116-8.

Weiss, U., Biber, P., Laible, S., Bohlmann, K., Zell, A., 2010. Plant species classification using a 3d lidar sensor and machine learning, in: 2010 Ninth International Conference on Machine Learning and Applications, pp. 339–345. doi:10.1109/ICMLA.2010.57.

Wong, S.C., Gatt, A., Stamatescu, V., McDonnell, M.D., 2016. Understanding data augmentation for classification: when to warp? CoRR abs/1609.08764. arXiv:1609.08764.

Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., Xiao, J., 2015. 3d shapenets: A deep representation for volumetric shapes, in: 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1912–1920. doi:10.1109/CVPR.2015.7298801.

Xiao, J., Owens, A., Torralba, A., 2013. Sun3d: A database of big spaces reconstructed using sfm and object labels, in: 2013 IEEE International Conference on Computer Vision, pp. 1625–1632. doi:10.1109/ICCV.2013.458.

Yuksel, C., 2015. Sample elimination for generating poisson disk sample sets. Computer Graphics Forum 34. doi:10.1111/cgf.12538.

Zhou, Q.Y., Park, J., Koltun, V., 2018. Open3d: A modern library for 3d data processing. `arXiv:1801.09847`.