Title:
**Point Cloud Dataset Creation for Machine Learning on CAD Models**

Authors:
Andrew R. Colligan, acolligan01@qub.ac.uk, Queen's University Belfast
Trevor T. Robinson, t.robinson@qub.ac.uk, Queen's University Belfast
Declan C. Nolan, d.nolan@qub.ac.uk, Queen's University Belfast
Yang Hua, y.hua@qub.ac.uk, Queen's University Belfast

Keywords:
Machine Learning, CAD Models, Point Cloud, Dataset Creation

DOI: 10.14733/cadconfP.2020.152-156

Introduction:
The application of machine learning (ML) is becoming common in many different fields. This is being referred to as a new digital revolution comparable to the invention of the internet; allowing for digital transformations in many different industries. Machine learning is the science of getting computers to learn to perform tasks without being explicitly programmed. The use of machine learning in Computer-Aided Design (CAD) modelling could have several applications. Examples of these include: the identification of machining features (i.e. slots and holes) for CAD to Computer-Aided Manufacturing (CAM) integration or the detection of features that hinder the creation of high quality Finite Element (FE) meshes, that normally must be manually removed before analysis can be undertaken. Although, there are several problematic issues that must be addressed. The first is that many common CAD model formats cannot be used as the direct input into a machine learning algorithm such as a neural network. These algorithms require a fixed input size for each piece of data across the entire dataset used to train and test the algorithm. However, for a CAD model represented as a boundary representation (B-Rep), as is the norm in many CAD systems, its format does not lend itself to a fixed sized data structure. Therefore, the CAD models need to be converted into a different representation for ML. The most common representations are voxels, point clouds or meshes. Each of these discretize the 3D space in which the CAD model exists. However, this results in a loss of information that could be useful for training the algorithm such as geometric information. This information could be represented as input features to the ML algorithm. These features are different from those usually defined in CAD systems, referring to any information/pattern in which the machine learning algorithm can learn. In this paper, the two different sets of features will be differentiated by referring to each as either "CAD features" and "ML features". There is a scarcity of published research into the connection to CAD models, instead most research only works on these secondary representations. It would be helpful if these geometric CAD features could be re-incorporated into the secondary representation, so that they could be used as additional ML features that could boost the ML algorithm's performance.

Main Idea:
*Problem of Representing 3D Models*
For the purpose of machine learning, CAD models produced in commercial CAD systems such as CATIA V5 [1] or Siemens NX [11] are normally converted from B-Rep to a secondary representation. Examples include: voxels, point clouds and meshes. A voxel representation is effectively a 3D image which has a regular structure similar to a 2D pixel grid, but with additional depth information. This

spatial occupancy enumeration has several different types, but the most common is a binary format where if a voxel lies within the shape it is denoted with a 1 and if it lies outside the shape it is denoted with a 0. One benefit of this approach is that it is easy to take ML algorithms optimized for 2D image tasks and re-implement them for learning tasks on 3D voxels. However, this representation is highly computationally expensive for higher resolutions. This is because the computational cost scales cubically with an increase in resolution. For example, a model with a resolution of 64x64x64 has 262,144 parameters if this resolution is doubled to 128x128x128 the number of parameters increases to 16,400,384 (6156% more). At higher resolutions, the time to train ML algorithms becomes impractical and can also result in an inability to fit the data in memory. A voxel representation is also considered to be a dense representation in which information is stored about the entire volume of the shape. Yet, B-Rep models only contain information on the boundaries. Therefore, for most solid shapes a voxel representation stores a lot of redundant information that does not benefit the learning problem at hand.

A point cloud represents a 3D model as a set of data points in 3D space. These points discretize the surface of the 3D model, where each point is represented as a set of coordinates (x, y, z). Additional dimensions can be used to store the surface normals and other local or global CAD features. 3D scanning technology often represents objects as point clouds where depth points are extracted from the external surface of the objects. To this end, many papers have reported on the utilization of point clouds for machine learning purposes. Although, initial papers [9][10] converted the point cloud data into voxel or other formats before inputting the data into a machine learning algorithm. This is due to point clouds having an irregular structure. Unlike with voxel data, previously developed ML algorithms optimized for image tasks are not easily able to be adapted for this kind of structure. The conversion to a voxel representation however creates overly voluminous data as well as generating quantization artefacts that obstruct the natural invariance of the data. Recent papers [6][7] have been able to implement the point cloud directly as an input. This allows for a reduction in the number of parameters needed to represent the 3D shape in comparison to voxel representations; as information is also only needed to be stored about the boundary of the shape. Therefore, this reduces the computational expense of training and implementing the algorithm.

Meshes are similar to point clouds as they both have an irregular structure. Previous papers [3][4] have used triangular surface meshes for representing the shape, meaning that only information about the boundary is stored. This representation in turn, has many of the benefits of point clouds from a reduction in the number of parameters to a flexibility over the resolution. One advantage compared to point clouds is its connectivity information that can be used to represent the underlying surface of the model.

For this paper, a point cloud representation was chosen due to mesh approaches having the difficulty of creating a fixed input structure. It was assumed that if an effective approach for labelling could be found for point clouds, it could be easily adapted for mesh data.

*Importance of Additional Geometric CAD Features*

Many of the previous papers on machine learning for point cloud data utilize data sourced from 3D scans. Therefore, there was no original CAD model to probe for additional ML features. This has resulted in most papers not utilizing CAD features that could give increased performance to the ML algorithm. To clarify ML algorithms, especially neural networks, benefit from large datasets containing thousands to millions of pieces of data and there is a direct correlation between the accuracy of the algorithm and the amount of training data used. Access to larger amounts of data is one of the main reasons for the current surge in the application of machine learning in many industries. It is not sufficient to just give a ML algorithm data and expect it to learn. The data must also contain the information in which the algorithm is expected to learn, which is represented in the form of a label. In a task called supervised learning, the ML algorithm makes a prediction on what it thinks the label should be, this is then compared to the true label. Through calculating a loss metric between the true and predicted label, the algorithm can be altered to help it better predict the data on a different iteration of training.

The problem arises with having insufficient amounts of data for training on CAD models. There does not exist large repositories of CAD models with labels that may want to be learnt such as machining features. Although, there has been work on compiling a range of CAD models into datasets for machine learning purposes [5][14], these still lack the necessary labels for CAD applications. The other option is to automatically generate CAD models allowing for better control over the labelling process [13]. The problem with this approach is that the models created are often highly rudimental, lacking in the complexity seen in actual CAD models produced by an engineer. There is also the risk of the programmer of the generator algorithm unintentionally introducing ML features in the data that the ML algorithm could become over reliant on for learning, or the models could lack important ML features. In cases like these, it can be said that the data distribution of the generated data is not the same as the distribution of the actual data and therefore the ML algorithm cannot generalize for new data that it sees. This results in low-performance accuracy and an inability to learn the necessary ML features for the data it will be presented.

Where there is a lack of data available for machine learning, additional hand-engineered ML features can be used to improve performance. These ML features are chosen based on prior knowledge of their importance to the problem. Effectively, instead of getting the ML algorithm to learn every feature from scratch, it is given a head start by providing extra important information. It is here that some CAD geometric information could give significant performance gains. It is therefore paramount that there are methods of extracting this geometric information for machine learning purposes.

*Point Cloud Generation*

Within CAD systems, there is often much functionality for importing point clouds, however there lacks functionality for point cloud generation. Ma et al [8] created point clouds from sampling the nodes in a mesh. The problem with this was that the mesh did not uniformly sample the 3D model. Instead, it produced dense regions in areas of higher detail. This artefact is generally desirable for Computer-Aided Engineering (CAE), the intended purpose of such meshing algorithms. However, the authors' future work will include the application of machine learning to learn to decompose CAD models for the generation of a hexahedral dominant mesh for analysis [14]. For such applications, this will leave areas of the CAD model insufficiently described, where new face splits would be created during the decomposition task and could be detrimental for the learning process.

In this paper, an external open-source program is chosen for the point cloud generation CloudCompare [2]. The CAD system used in the experiments was Siemens NX. For the importation of the CAD model into the point cloud generation program, it requires the discretization of the model into a mesh. This is achieved by creating a Standard Triangle Language (STL) version of the model. A point cloud is then generated with a specific size e.g. 10,000 points. This is then imported back into the CAD system to be labelled. Fig. 1 shows a general process flow for the labelling of a point cloud with information from a decomposition task. The reason for the generation of the point cloud from the original undecomposed CAD model is due to the fact that new geometry is created in the decomposition process. This will not be present for a non-decomposed model given to the ML algorithm after training. Therefore, the algorithm needs to not be reliant on this new information generated in the decomposition task. This approach allows for this to be apparent and enables the gathering of both non-decomposed and decomposed geometric information. By sampling uniformly and performing a study on the number of points used to represent the data accurately, one is also able to be sure that all regions are sufficiently described.

*Point Cloud Labelling*

One of the most important aspects of this process, is the ability to gage which face each point belongs to. The identification of these CAD face features allows for easier transversal of the model to obtain other CAD features such as edges or vertices. The first step in this labelling process is to superimpose the point cloud onto the CAD model. One problem that arises from the conversion of the B-Rep to an STL is that points that should lie on curved faces no longer do. This means that one cannot simply search if the point is contained within a face to label it.

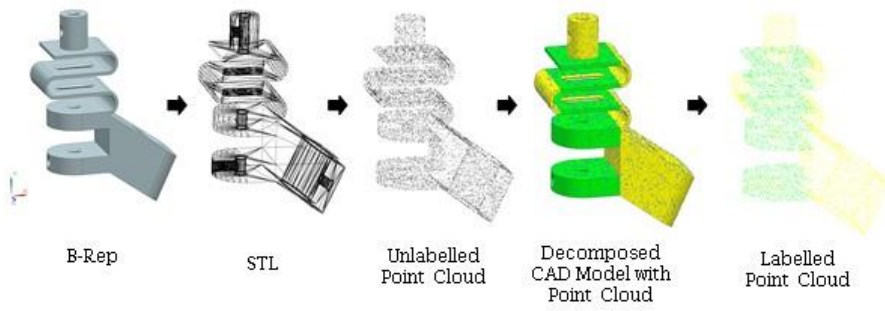


Fig. 1: Process Flow of Point Cloud Labelling for a Decomposition Task.

Although, the distance of the point from the face can be used to indicate if it belongs to that face. It would be computationally expensive to calculate the distance between every point in the point cloud and every face in the model. Instead, a bounding box is initially calculated for every face in the model. Then, for a point it can be checked whether it lies in any of the bounding boxes. The check is done across all the bounding boxes at once using fast array operations from the C# Linq library. This in turn, decreases the number of potential faces in the search space on average by 97.89% for a point cloud of 10,000 points. A tolerance is added to the bounding box, due to some points no longer lying on the surface, meaning they may not be contained within the exact bounding box. The tolerance heuristically is chosen to be 0.1mm. For all the potential faces, the distance metric is calculated and then the face with the smallest distance is chosen. If the distance metric calculated is 0 then this face is assumed to be the correct face. Fig. 2 shows the algorithm used to label the point cloud.

**Input:** Work part & unlabelled point cloud
**Output:** Labelled point cloud with assigned faces

```
1   foreach body in part
2     foreach face in body
3         Calculate bounding box for face
4         Add tolerances to bounding box
5         Store bounding box in dictionary with coordinates and
          the corresponding face tag
6   foreach point in point cloud
7     Check which bounding boxes the point lies within
      This operation will return the face tags of the bounding boxes
8     foreach face in filtered faces
9         Calculate the minimum distance between face and point
10        if minimum distance to face is 0
11            Assign face to point
12        else
13            Store minimum distance in array
14    Find the smallest distance in minimum distance array and
      assign corresponding face to point
```

Fig. 2: Algorithm for Face Assignment for Point Clouds.

Conclusions:

The following conclusions have been drawn for this work:

- There is a need for methodologies for creating fixed size datasets enriched with CAD model information (e.g. CAD features, face properties), suitable for use with machine learning algorithms.

- A complete pipeline for the creation and labelling of point cloud data from CAD models is presented.
- An algorithm for the assignment of face labels to point clouds has been developed with the limitations addressed.

Acknowledgements:
Author Andrew R. Colligan is a PhD researcher who is funded through DfE government funding.

References:
[1] CATIA V5, https://www.3ds.com/products-services/catia/
[2] Cloudcompare, https://www.cloudcompare.org/
[3] 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, 2019, 8279–86. https://doi.org/10.1609/aaai.v33i01.33018279
[4] Hanocka, R.; Hertz, A.; Fish, N.; Giryes, R.; Fleishman, S.; Cohen-Or D.: MeshCNN: A Network with an Edge, 2018. https://doi.org/10.1145/3306346.3322959
[5] Koch, S.; Matveev, A.; Jiang, Z.; Williams, F.; Artemov, A.; Burnaev, E.; Alexa, M.; Zorin, D.; Panozzo, D.: ABC: A Big CAD Model Dataset For Geometric Deep Learning, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019. https://doi.org/10.1109/CVPR.2019.00983
[6] Qi, C. R.; Li, Y.; Hao, S.; Guibas, L. J.: PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space, NIPS'17: Proceedings of the 31st International Conference on Neural Information Processing Systems, 2017, 5105-5114
[7] Qi, C. R.; Su, H.; Kaichun, M.; Guibas, L. J.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, 77–85. https://doi.org/10.1109/CVPR.2017.16
[8] Ma, Y.; Zhang, Y.; Luo, X.: Automatic Recognition of Machining Features Based on Point Cloud Data using Convolution Neural Networks, AICS 2019: Proceedings of the 2019 International Conference on Artificial Intelligence and Computer Science, 2019, 229–35. https://doi.org/10.1145/3349341.3349407
[9] Maturana, D.; Scherer, S.: VoxNet: A 3D Convolutional Neural Network for real-time object recognition, In: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2015, 922–8. https://doi.org/10.1109/IROS.2015.7353481
[10] Sedaghat, N.; Zolfaghari, M.; Amiri, E.; Brox, T.: Orientation-boosted voxel nets for 3D object recognition, British Machine Vision Conference (BMVC), 2017. https://doi.org/10.5244/C.31.97
[11] Siemens NX, https://www.plm.automation.siemens.com/global/en/products/nx/
[12] Sun, L.; Tierney, C.; Robinson, T.; Armstrong, C.: Automatic decomposition of complex thin walled CAD models for hexahedral dominant meshing, Procedia Engineering, 2016, 163. https://doi.org/10.1016/j.proeng.2016.11.052
[13] Zhang, Z.; Jaiswal, P.; Rai, R.: FeatureNet: Machining feature recognition based on 3D Convolution Neural Network, Computer-Aided Design, 101, 2018, 12–22. https://doi.org/10.1016/j.cad.2018.03.006
[14] Wu, Z.; Song, S.; Khosla, A.; Fisher, Y.; Zhang, L.; Tang, X.; Xiao, J.: 3D ShapeNets: A deep representation for volumetric shapes, 2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2015, 1912–20. https://doi.org/10.1109/CVPR.2015.7298801