







Point Cloud Dataset Creation for Machine Learning on CAD Models

Andrew R. Colligan¹ , Trevor T. Robinson² , Declan C. Nolan³  and Yang Hua⁴ 

¹Queen's University Belfast, acolligan01@qub.ac.uk

²Queen's University Belfast, t.robinson@qub.ac.uk

³Queen's University Belfast, d.nolan@qub.ac.uk

⁴Queen's University Belfast, y.hua@qub.ac.uk

Corresponding author: Trevor T. Robinson, t.robinson@qub.ac.uk

Abstract. Recently, the application of machine learning on Computer-Aided Design (CAD) models has emerged. However, there is a lack of robust methods for the conversion of boundary representation (B-Rep) CAD models from engineering software to appropriate input representations for a machine learning algorithm. Those that do exist break the link with the B-Rep, meaning the ability to use machine learning to support future engineering operations on the B-Rep are challenging. This paper presents a method for the creation and labelling of point clouds from B-Rep CAD models for machine learning techniques, while maintaining a link between the two representations. This method allows for the creation of a dataset with additional input features determined from the CAD model such as B-Rep face labels, that could increase the accuracy of certain problems when machine learning is utilized. First, an open-source software called CloudCompare is used for point cloud creation. Fast interrogation of the CAD model using face bounding boxes are utilized to link points to their corresponding faces. This link allows for easy traversal of the CAD model topology to gain other geometric features if needed. A deficiency of the approach is that some B-Rep faces result in being under sampled. For these insufficiently sampled faces, a method is presented to re-sample them. Experiments on the approach are outlined to illustrate the efficiency of the proposed method with the approach taking approximately 10 seconds per CAD model within the tested dataset.

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

DOI: <https://doi.org/10.14733/cadaps.2021.760-771>

1 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 [1]. 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 an input of fixed size for each piece of data across the entire dataset used to train and test the algorithm. However, for CAD models represented as a boundary representation (B-Rep), as is the norm in most commercial 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 and meshes. Each of these use different methods to 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" or "ML features". There is a scarcity of published research into the connection to B-Rep CAD models, instead most research only works on these secondary representations. For engineering purposes, 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.

2 RELEVANT WORK

2.1 Problem of Representing 3D Models

For the purpose of machine learning, CAD models produced in commercial CAD systems such as CATIA V5 [3] or Siemens NX [15] are normally converted from B-Rep to a secondary representation [6][18]. Examples include: voxels, point clouds and meshes, as defined below.

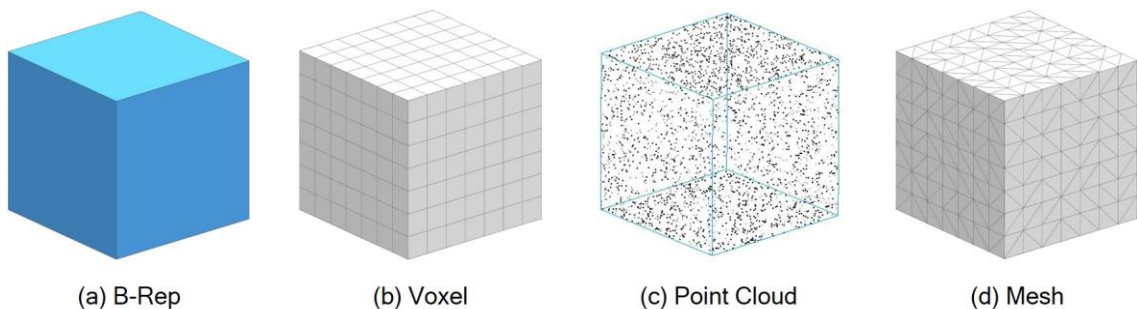


Figure 1: Illustrations of B-Rep and Secondary CAD Model Representations.

A B-Rep as seen in Figure 1(a), consists of geometry (e.g. surfaces, curves and points) linked together through topology (faces, edges and vertices). A voxel representation as shown in Figure 1(b) is effectively a 3D digital 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. Otherwise, it is denoted with a 0. One benefit of this approach is that it is relatively 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. In contrast, 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 considered to be a dense representation as information is stored for the entire volume of the shape. In contrast, B-Rep models only contain information on the boundaries. Therefore, for most solid shapes, a voxel representation stores a lot of redundant information on the interior 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, Figure 1(c). 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 techniques. Although, initial papers [13][14] converted the point cloud data into voxel or other formats before applying machine learning algorithms. This is due to point clouds having an irregular structure. Unlike with voxel data, previously developed ML algorithms optimized for image tasks cannot be easily adapted for this kind of structure. The conversion to a voxel representation creates overly voluminous data as well as generating quantization artefacts that obstruct the natural invariance of the data. Recent papers [10][11] 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 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 in that they both have an irregular structure. Previous papers [5][7] have used triangular surface meshes as seen in Figure 1(d) for representing the shape meaning that only information about the boundary is stored. As a result, this representation 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.

2.2 Importance of Additional Geometric CAD Features

Many of the previous papers on machine learning with point cloud data utilize data sourced from 3D scans. Therefore, there was no original B-Rep CAD model to probe for additional ML features. This has resulted in most papers not utilizing feature information provided by CAD models 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 where 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 enough to just give data to a ML algorithm and expect it to learn properly. The data must also contain sufficient information through 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 predicts what it thinks the label should be, which 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 B-Rep CAD models. There does not exist large repositories of B-Rep CAD models with suitable labels to be learnt such as machining features. Although, there has been work on compiling a range of CAD models into datasets for machine learning techniques [9][17], they still lack the necessary labels for engineering applications. The other option is to automatically generate CAD models allowing for better control over the labelling process [18]. The problem with this approach is that the models created are often highly rudimentary, 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 such cases, 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 from the data.

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.

3 POINT CLOUD PROCESSES

3.1 Point Cloud Generation

Within CAD systems, there is often functionality for importing point clouds, however, there lacks functionality for point cloud generation. Ma et al [12] 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 [16]. For such decomposition applications, it will leave areas of the CAD model insufficiently described, where new splitting faces would be created during the decomposition task and could be detrimental for the learning process.

In this paper, an external open-source program CloudCompare [4] is chosen for the point cloud generation. 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. Figure 2 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.

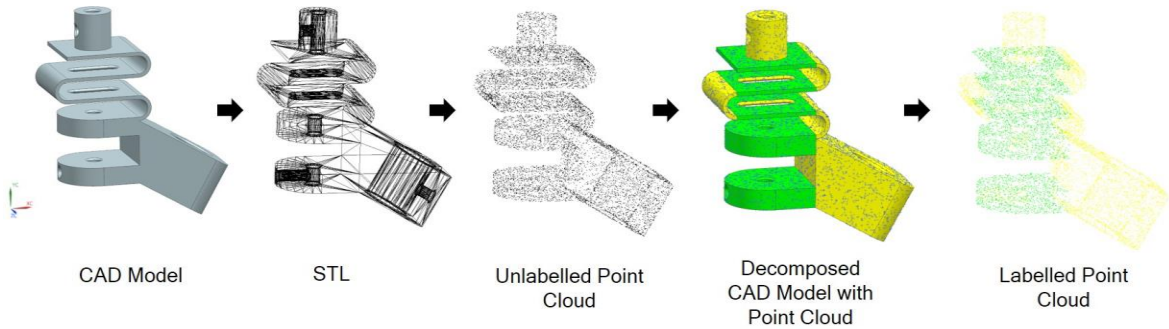


Figure 2: Process Flow of Point Cloud Labelling for a Decomposition Task.

3.2 Point Cloud Labelling

One of the most important aspects of this process, is the ability to gauge which geometric face each point belongs to. The identification of these CAD face features allows for easier traversal 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 are no longer guaranteed to. This means that one cannot simply search if the point is contained within a face to label it.

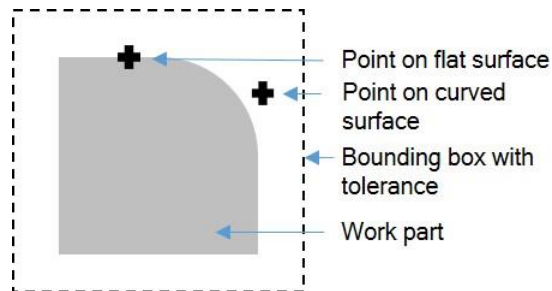


Figure 3: Bounding Box for Work Part.

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, it can be checked whether a point 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# Language Integrated Query (LINQ) library [2]. A tolerance is added to the bounding box as can be seen in Figure 3. This is 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. Information on the selection of this metric can be found in Section 4.1.

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 0mm then this face is assumed to be the correct face. Figure 4 shows the algorithm used to label the point cloud. Figure 5 outlines the steps within the algorithm for selecting the corresponding face of a point in 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

```

Figure 4: Algorithm for Face Assignment for Point Clouds.

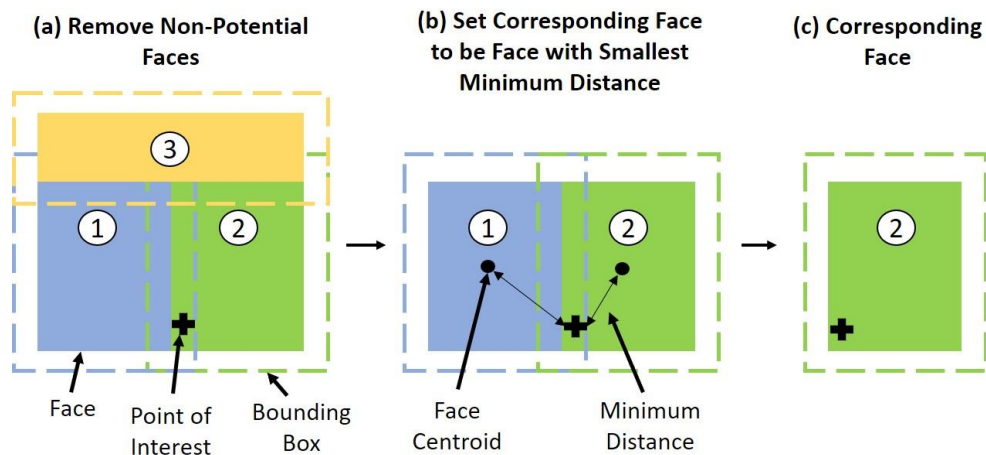


Figure 5: Illustration of the Process to Select the Corresponding Face for a Point.

3.3 Supplementing Non-Sampled Faces

One issue that may appear is if the sampled point cloud does not contain a point on every face in the CAD model. This arises from two situations. The first is if the CAD model contains relatively small faces in comparison to the other faces found in the CAD model. In this situation, the CloudCompare software may miss these faces when sampling the point cloud. The second situation is where the sampled CAD model is further decomposed for meshing purposes, therefore creating additional faces as can be seen in Figure 6. These faces may not be sufficiently sampled by the original point cloud. Therefore, the ability to identify and re-sample these faces would be greatly beneficial. Figure 7 shows an example where small faces within the CAD model were insufficiently sampled by the generated point cloud.

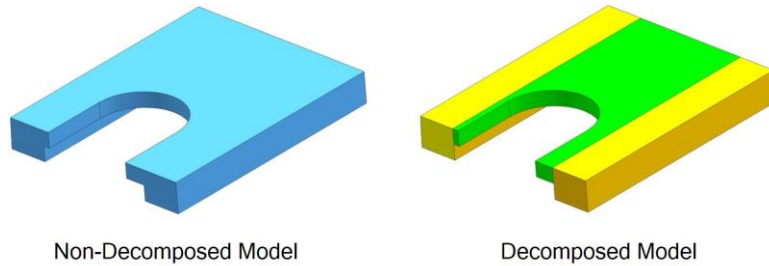


Figure 6: Creation of New Geometric Faces from Decomposition.

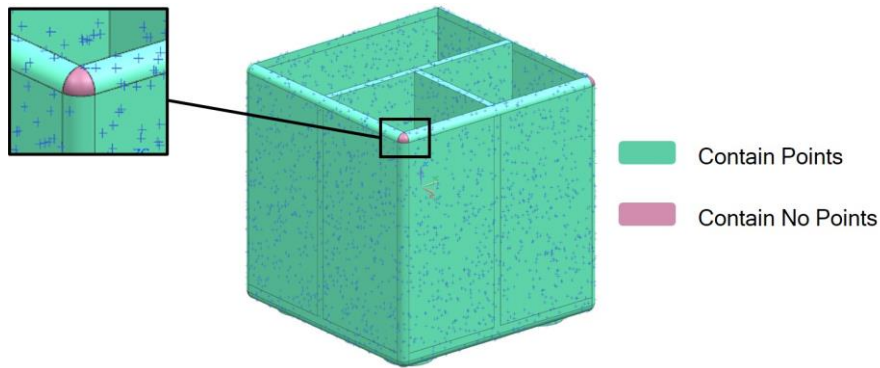


Figure 7: Example of Insufficiently Sampled Faces.

For CAD models that contain only one body, the insufficiently sampled faces can be found by simply querying which faces contain no points. The problem occurs for models where multiple bodies exist. An example of this would be for CAD models which have been decomposed for meshing. In these types of CAD models, there exists internal faces between the bodies which should not be sampled. There are two types of potential internal faces. Figure 8(b) shows the sheet bodies used to split the original CAD model for decomposition. Each of these bodies contain faces that can be easily identified by probing the sheet bodies. The red faces in Figures 8(c) & 8(d) show the internal faces created by the split body operation that belong to the solid bodies. For each location where two bodies meet there will exist at least two of these faces. Within the Siemens NX API, it is not clear how to identify these faces with any existing function.

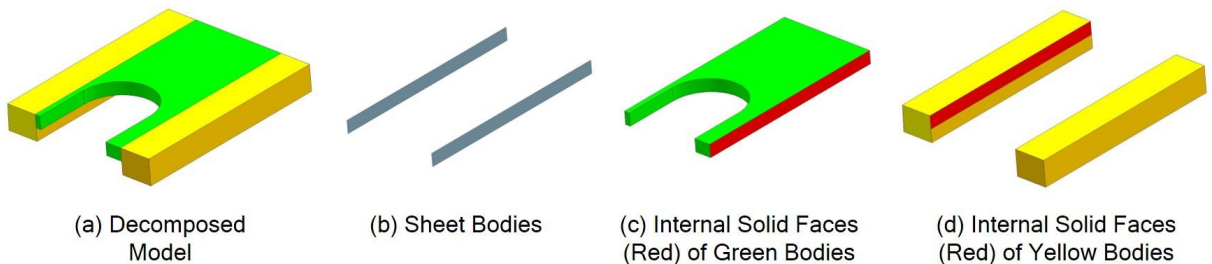


Figure 8: Internal Sheet and Solid Faces Produced for Decomposition.

To identify the internal faces of the solid bodies, the faces of the sheet bodies can be used. This is because these sheet faces should have the same imprint as the solid internal faces. As the imprints should be similar, the centroids and areas of these two types of faces should also be comparable. Therefore, by matching the solid faces with centroids and areas similar to those of the sheet faces, the internal solid faces can be identified. However, searching this potential face space could be very computationally expensive. To speed up the search, a k-d tree is utilized [8]. A 3-dimensional tree is built using the coordinates of the centroids of all the faces of solid bodies in the CAD model. For each face in the sheet bodies, the k-d tree is searched to find the two closest neighbors for their centroids. The area of these potential internal solid faces is then compared to the area of the sheet face. A tolerance is added to the sheet face area for situations where the area of the internal face is identified as being slightly different.

One situation where it has been identified that this method will fail is where the splitting operation creates multiple new faces on the solid body from one sheet face. In this case, the areas of the solid body and sheet body faces will not match. An example of this is shown in Figure 9. This could be potentially solved by widening the number of neighboring centroids to test, then with any unlabeled centroids create a set of unique permutations of faces. Then, the areas of these face permutations could be summed and compared to the area of the sheet face. Although, this would be immensely computationally intensive. Another approach would be to perform a union Boolean operation on the bodies while maintaining the existing external faces. However, this functionality is not obvious in NX, instead, some of the external faces will be merged in the process.

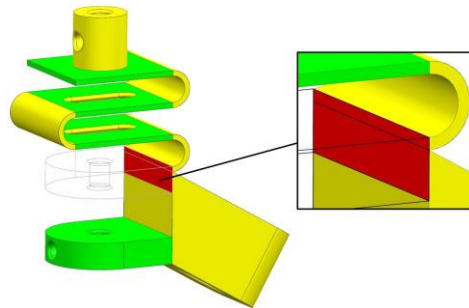


Figure 9: Missed Internal Solid Faces.

Once the external and internal solid faces have been identified, the ratio of the number of points to face area is calculated for the external solid faces. If any faces have a ratio of zero, then they are re-sampled. This is achieved by randomly removing points from the faces that have the largest ratio, therefore ensuring that the number of points in the point cloud remains the same. A minimum number of points per face is also applied which is user-defined, in case faces with small areas lack adequate number of points to describe them. The scope of re-sampling could also be expanded to faces with low ratios; however, this would increase the computational expense. A method was also tested where new points were created and the point cloud size was altered, with the assumption that the point cloud could later be randomly downsampled to achieve the desired size. Once each point has been associated with a face in the B-Rep model, then the dataset used for ML can be bolstered with attributes from the B-Rep (e.g. grouping points by B-Rep face).

4 EXPERIMENTS

In this section, several experiments are detailed which were undertaken to either help in the selection of parameters for the approaches or to measure their effectiveness. For each experiment, a dataset of 1,000 CAD models was used. The models used for this dataset were randomly selected from the ABC dataset [9].

4.1 Tolerance Selection

As already stated, an exact bounding box of a face may not contain points that are created from the discretization of curved surfaces, a tolerance is needed to be added to the dimensions of the bounding boxes. In order to select an appropriate tolerance value, an experiment was undertaken to find the max and average distance of points from their corresponding face. To achieve this, each point in the point cloud was probed to find its distance from each face in the CAD model. The face with the smallest distance to the point was assumed to be the corresponding face. This distance value was recorded for each point in the point cloud. The average and maximum distance of this set were calculated for each model in the CAD dataset. Table 1 shows the results for the entire dataset, where the point clouds consist of approximately 10,000 points.

	<i>Average Face Distance (mm)</i>	<i>Maximum Face Distance (mm)</i>
Minimum Outlier	0	0
Quartile 1 (25%)	7.01E-05	5.03E-03
Quartile 2 (50%)	8.23E-04	7.88E-03
Quartile 3 (75%)	1.79E-03	8.19E-03
Maximum Outlier	5.14E-03	2.57

Table 1: Average and Maximum Face Distance Calculations for CAD Dataset.

Table 2 shows how the search space is reduced by the use of different tolerances on the face bounding boxes.

<i>Tolerance (mm)</i>	<i>Avg. Faces in Search Space</i>	<i>Reduction of Search Space (%)</i>
No Tolerance	104.18	0
2.6	5.74	94.53
1	3.97	96.22
0.1	2.21	97.89
0.01	1.98	98.11

Table 2: The Effect of Altering Tolerance on the Face Search Space.

Table 1 shows that on average the furthest distance a point will lie from a face is 5.14E-03mm. With 75% of the maximum face distances lying within 8.19E-03mm. Although, there are some outliers with the furthest distance from a face being 2.57mm. From Table 2, it can be seen that a tolerance of 0.01mm and 0.1mm have a similar reduction in the search space with a difference between the two of 0.22%. Therefore, a tolerance of 0.1mm is used in this paper. As this would have similar efficiency to a tolerance of 0.01mm, while still hopefully encapsulating some of the points that are outliers.

4.2 Comparison of Point Cloud Sizes

To understand the scalability of the method, the basic approach without face re-sampling was run with different point cloud sizes. Two different cases were tested: the first is non-decomposed models where the original models from the ABC dataset were used. The second is decomposed CAD models. Four different point cloud sizes were analyzed: 5,000, 10,000, 20,000 and 40,000 points. As shown in Figure 10, the approach scales quadratically with an increase in the number of points. Therefore, it could be assumed that point cloud sizes above those analyzed would be highly expensive to label. However, the author is not aware of any published machine learning research, that has utilized point clouds of that density. Therefore, the current approach can be deemed appropriate. There is also a slight increase in the time taken to label the decomposed CAD models. This can be attributed to the increased number of faces in the CAD model.

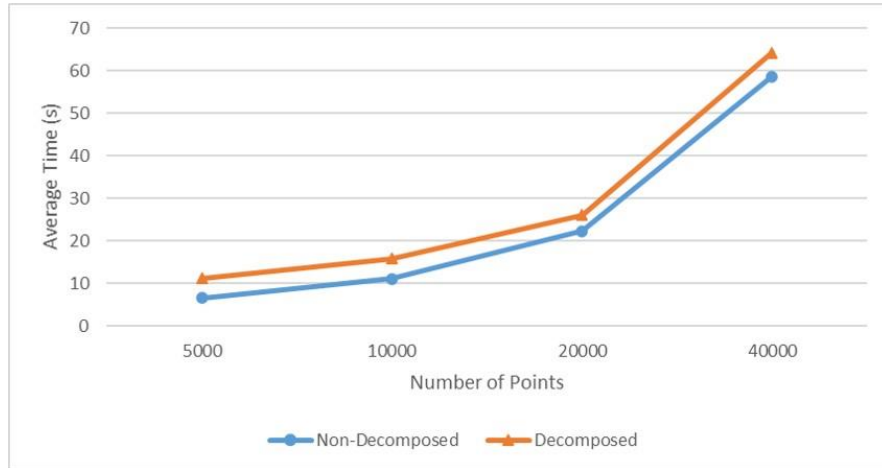


Figure 10: Effect of Varying Point Cloud Size on the Time for Operation.

To ensure that the point cloud was sufficiently sampling the faces of the CAD model, the number of non-sampled faces was found for each of the CAD models in the dataset. This was compared to the total number of faces in the CAD model. This was only conducted on the decomposed CAD models. Table 3 highlights the averages for the dataset and how they change for different point cloud sizes.

<i>Point Cloud Size</i>	<i>Average Number of Non-Sampled Faces</i>	<i>Average % of Non-Sampled Faces to Total Faces</i>
5,000	18	6.16
10,000	14	4.65
20,000	10	3.46
40,000	6	2.49

Table 3: Average Ratio of Number of Points per Face Area and Number of Non-Sampled Faces.

Intuitively, Table 3 shows that as the point cloud size increases, the number of un-sampled faces decreases. This roughly attributes to, if the point cloud size doubles, the percentage of non-sampled faces to total faces decreases by 25-28%. Overall, it can be said that the point clouds created by CloudCompare are able to describe the majority of faces in the CAD model, with the sparsest point cloud size still achieving 6.16% for its non-sampled face to total faces metric. Therefore, this approach seems to be able to describe new faces created from a decomposition operation.

4.3 Comparison of Approach Versions

Lastly, the effect of using different versions of the approach was compared. For this only the decomposed models were used, and the point cloud size was kept at approximately 10,000 points. Three different versions were compared. The first only used bounding boxes, the second used the bounding boxes and re-sampled faces but did not relocate points, and the last used the bounding box and re-sampling by way of relocating points. Table 9 shows the comparison between each version. It can be seen that the point relocation version was over 4 times slower than the other versions. This may be down to how the algorithm may select a point that has already been sampled. In this case, the algorithm will need to loop through to find another suitable point. The basic bounding box and new point creation versions are almost equal in terms of runtime. Therefore, it can be assumed that the k-d tree method for finding the internal solid faces is not the bottleneck of the process.

<i>Version</i>	<i>Average Time per CAD Model (secs)</i>
Basic Bounding Box	10.31
New Point Creation	10.48
Point Relocation	48.48

Table 4: Average Ratio of Number of Points per Face Area and Number of Non-Sampled Faces for Different Versions of the Approach.

5 CONCLUSION

The following conclusions have been drawn for this work:

- 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. This achieves an average time per CAD model of approximately 10 seconds.
- A version of the approach to re-sample faces which are insufficiently described by the point cloud is also presented, with the fastest version having an average time per CAD model of 10.48 seconds.
- There is an exponential increase in the time taken to create the point cloud and the number of points used to sample the CAD model.

6 FUTURE WORK

The authors' future work will include the application of machine learning to learn to decompose CAD models for the purpose of hexahedral mesh generation. Other future avenues of work could include the normalization of the CAD model scale to produce an adaptive bounding box tolerance. And the movement of points back onto geometric faces where they have been moved during the STL creation process.

7 ACKNOWLEDGEMENTS

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

Andrew R. Colligan, <https://orcid.org/0000-0002-7904-5644>

Trevor T. Robinson, <https://orcid.org/0000-0002-6595-6308>

Declan C. Nolan, <https://orcid.org/0000-0002-9388-6183>

Yang Hua, <https://orcid.org/0000-0001-5536-503X>

REFERENCES

- [1] Andrew Ng's Machine Learning course, <https://www.coursera.org/learn/machine-learning>
- [2] C# LINQ, <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/linq/>
- [3] CATIA V5, <https://www.3ds.com/products-services/catia/>
- [4] CloudCompare, <https://www.cloudcompare.org/>
- [5] 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>

- [6] Ghadai, S.; Balu, A.; Sarkar, S.; Krishnamurthy, A.: Learning localized features in 3D CAD models for manufacturability analysis of drilled holes, *Computer Aided Geometric Design*, 62, 2018, 263-275. <https://doi.org/10.1016/j.cagd.2018.03.024>
- [7] Hanocka, R.; Hertz, A.; Fish, N.; Giryas, R.; Fleishman, S.; Cohen-Or D.: MeshCNN: A Network with an Edge, 2018. <https://doi.org/10.1145/3306346.3322959>
- [8] K-d trees, https://pcl-tutorials.readthedocs.io/en/latest/kdtree_search.html#kdtree-search
- [9] 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>
- [10] 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. <https://dl.acm.org/doi/10.5555/3295222.3295263>
- [11] 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>
- [12] 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>
- [13] 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>
- [14] 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>
- [15] Siemens NX, <https://www.plm.automation.siemens.com/global/en/products/nx/>
- [16] 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>
- [17] 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>
- [18] 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>