Deep Learning Classification in web3D model geometries

Using X3D models for Machine Learning Classification in Real-Time web applications

Chrysoula Tzermia
Dept. of Electrical and Computer Engineering, Hellenic Mediterranean University
chrisajerm@gmail.com

Nick-Periklis Chourdas
Dept. of Electrical and Computer Engineering, Hellenic Mediterranean University
nikos.hourdas@gmail.com

Athanasios G. Malamos
Dept. of Electrical and Computer Engineering, Hellenic Mediterranean University
amalamos@hmu.gr

ABSTRACT

In this paper we study about the requirements of web3D models and particular X3D formatted models in order to work efficiently with Deep Learning algorithms. The reason we are focusing in this particular type of 3D models is that we consider web3D as part of the future in computer graphics. The introduction of metaverse™ technology, indeed confirms that lightweight interoperable 3D models will be an essential part of many novel services we will see in the near future. Furthermore, X3D language is expressing 3D information in a way semantically friendly and so very useful for future applications. In our research we conclude that the lightweight X3D models require some vertices enhancement in order to cooperate with Deep Learning algorithms, however we suggest algorithms that may be applied and make the whole process in Real-Time which is very important in case of web applications.

CCS CONCEPTS

• **Computer graphics;**; • **Graphics systems and interface;**; • **Perception**;

KEYWORDS

Machine Learning, Deep Geometry Learning, Web3D models machine learning classification

ACM Reference Format:

Chrysoula Tzermia, Nick-Periklis Chourdas, and Athanasios G. Malamos. 2022. Deep Learning Classification in web3D model geometries: Using X3D models for Machine Learning Classification in Real-Time web applications. In *The 27th International Conference on 3D Web Technology (Web3D '22), November 02–04, 2022, Evry-Courcouronnes, France.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3564533.3564564

1 INTRODUCTION

Machine learning classification is a rather mature research area that gains more and more attention during the last decade especially after the introduction of Deep Learning technology. The use of Deep Learning algorithms in geometry classification has also gained some serious attention [Guo 2021; Zhou 2020 ; Zhou 2021] since it is related to the classification and finally recognition of geometries and objects that are either synthetic or even better created by the use of sensors like cameras (ex. photogrammetric techniques) or beam scanners (ex. laser, sonar, infrared etc.). The special case of sensor created 3D models has gained even more attention the last five years, since creating point clouds through sensors became simpler, processors and smart devices are becoming more and more powerful and cloud technology makes feasible to use complicated deep learning algorithms in real time.

However, point clouds are not fully 3D models with vertices and polygons, but rather are just a number of distributed points (vertices). Creating polygons out of points is and additional and rather complicated process that increases complexity and usually makes the process hard non-real-time. Thus, community focuses on classification algorithms that work directly with the point cloud and extraction of skeletal and surface characteristics are considered as part of the Deep Learning algorithm and the corresponding training process.

In this paper we focus on classification of web3D models and particular X3D formatted models. The reason we are focusing in this particular type of 3D models is that we consider web3D as part of the future in computer graphics. The introduction of metaverse technology, indeed confirms that lightweight interoperable 3D models will be an essential part of many novel services we will see in the near future. Furthermore, X3D language is expressing 3D information in a way semantically friendly and so very useful for exploitation.

In [Feng et al., 2018] authors present a deep learning algorithm that is appropriate for classifying meshes of 3D models. This method aggregates face properties of meshes and classifies models according to features extracted by deep learning training. The algorithm is trying to combine spatial features of the meshes (vertices and center point of a face) with structural such as normal and neighbor faces indexing. In fact we consider this algorithm as a fusion algorithm that engages features of point based methods with some features of the mesh to improve the lack of points that usually have in the case of mesh representations.

In [Kim 2020], authors present a methodology on how to classify X3D models using polygons and Deep Neural Network model algorithm. Authors do not present the exact algorithm they use or enough details about implementation. The presented results look very corroborative for the methodology although they do not specify whether they use a formal dataset for learning and testing the algorithm or not. Moreover, the implementation is considering running upon the polygons that makes preparation of the target models a rather complicated process especially if they are point clouds.

Web3D '22, November 02–04, 2022, Evry-Courcouronnes, France

ACM ISBN 978-1-4503-9914-2/22/11.
https://doi.org/10.1145/3564533.3564564

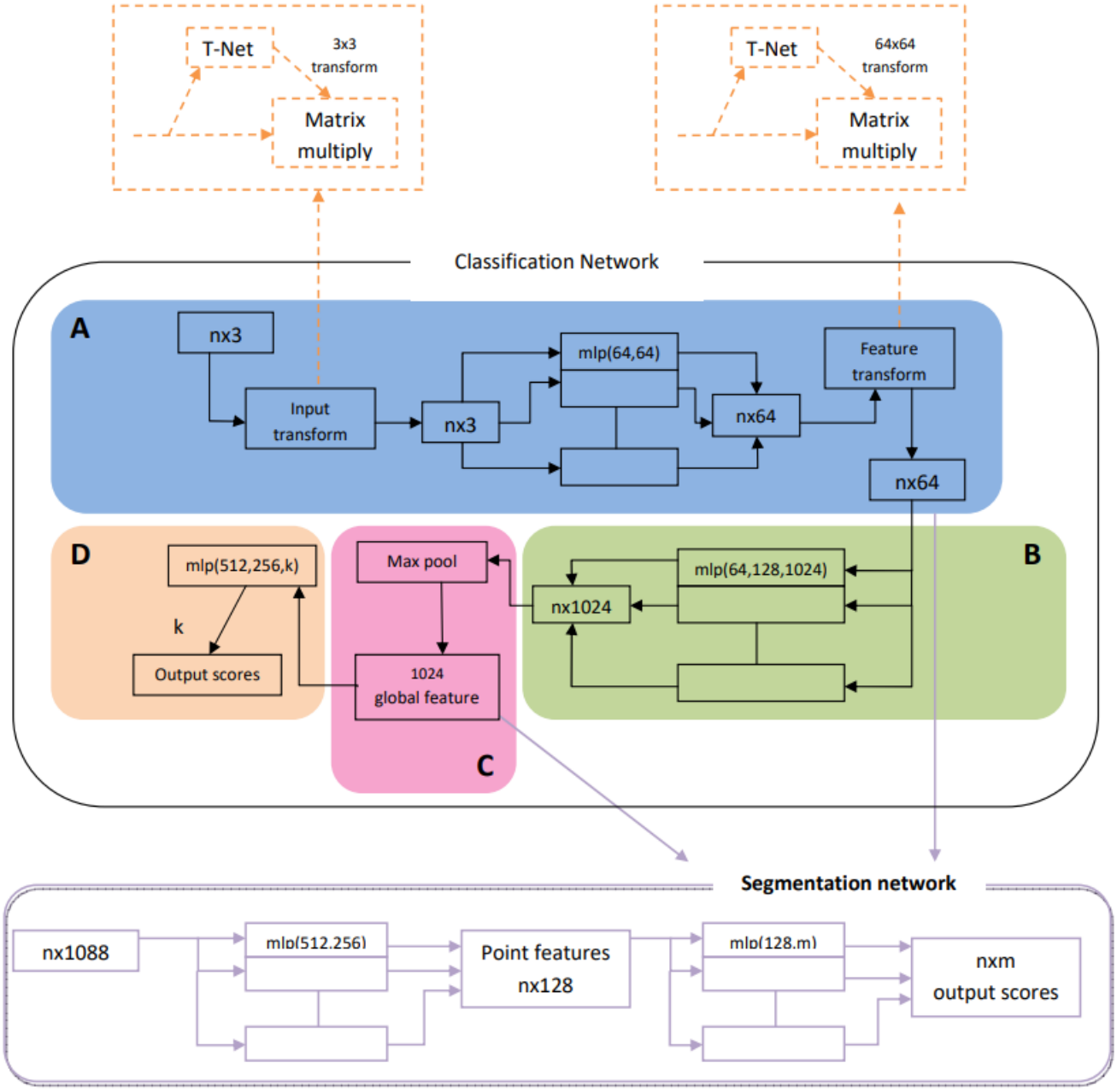


Figure 1: The PointNet Architecture Diagram

In our approach we are using PointNet [Qi 2020] a Deep Learning algorithm for unordered point clouds classification that makes the algorithm ideal for models generated by sensors. For the model description we use X3D IndexedFaceset with coord attribute to give us the point cloud. Since PointNet doesn't require any knowledge of surfaces of the 3D models, coordIndex in X3D IndexedFaceSet are only considered when is required resampling of the point clouds to increase efficiency. The problem that arises when using an point based algorithm in classification of meshes is that usually meshes doesn't provide enough points for the algorithm to work efficiently. We solve this problem by increasing the number of points to the required one by interpolating points inside faces of the mesh. Moreover we evaluate our concept using the ModelNet40 [PRINCETON 2022] with 40 individual classes of 3D objects. An important notice here is that the training set is not required to be in X3D format, only the testing set is in this format. The whole implementation of Deep

Learning is in Google ™ TensorFlow with KERAS environment and after training the algorithm is extracted in JSON format and runs in client-side inside webpage with TensorFlow Javascript runtime library.

We consider this paper as part of the tetralogy consisted of, point cloud segmentation, model extraction, model classification and finally semantic description. In this paper we focus on model classification. We are also working on segmentation and model extraction and we hope that we will present our results in a forthcoming paper. Moreover, previous works [Flotyński 2020] [Flotyński 2019] of the community deal with the semantic expression of 3D scenes and especially with semantical characterization of spatial interrelations between individual objects inside a scene.

The rest of this manuscript is structured as follows: in section 2 we present the PointNet algorithm and we are providing several details about the implementation, in section 3 we deal with the required real-time conditioning of the X3D models point clouds in order to increase efficiency of the methodology, in section 4 we present the results of the experiments with the ModelNet40 dataset and original X3D scenes in webpages, in section 5 we present our conclusions for the implementation of the methodology as well as future work..

2 THE DEEP LEARNING ARCHITECTURE

The architecture we use is mainly based on the PointNet algorithm. PointNet is an innovative, highly efficient net that uses neural networks to detect 3D objects without rendering. It was created by the team of Stanford University [Qi 2020] to provide several applications for scene semantic parsing to object classification.

The basic idea of this algorithm for classification and segmentation of unordered point clouds is to estimate the distance between points. This allows us to condense points that are close together by grouping them in small "boxes". This method may be used to summarize geometric information and eventually label the complete point neighborhood.

The diagram in Figure 1 shows the architecture of PointNet. More specifically the classification network maps each of the n points from 3 dimensions to 64 dimensions by using a shared multi-layer perceptron. Each of the n points has its own multi-layer perceptron (A on diagram). Similarly, each n point is transferred from 64 to 1024 dimensions in the following layer (B on diagram). A max pooling is further used to construct a Global Feature Vector in R^1024 (C on diagram). Finally, the Global Feature Vector is mapped to k output classification scores using a three-layer fully connected network (FCN) (D on diagram). Each of the n input points in the segmentation network must be allocated to one of k segmentation classes. Segmentation relies on both local and global features, the points in the 64-dimensional embedding space (local point features) are concatenated with the global feature vector (global point features) to produce a per-point vector in R^1024 . In other words, after computing the global feature vector, the algorithm feeds it back to the point feature by concatenating global features with per point features to get the desired outcome. This approach can anticipate per-point quantities by relying on both global and local semantics. MLPs are used on the n points to reduce the dimensionality from 1088 to 128 and subsequently to m, resulting in an array of n x m.

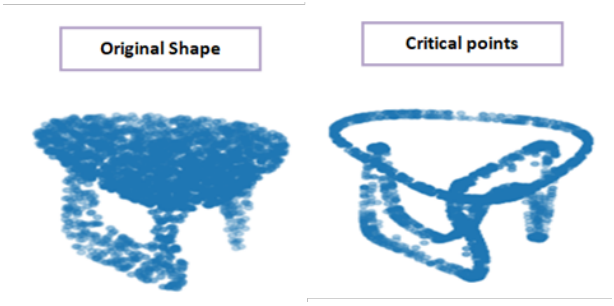


Figure 2: PointNet extracts the critical points (skeleton) of the cloud.

The Global Feature Vector may be used to derive a significant amount of intuition. To begin, as previously stated, the dimensionality of the vector, referred to as the critical dimension and denoted by K, is directly related to the expressiveness of the model. Naturally, a higher K value leads to a more complicated — and, more importantly, correct — model, and vice versa. K=1024, for example, is used in the design of PointNet. Also keep in mind that the feature vector was the outcome of a well-thought-out symmetric function (for permutation invariance). PointNet employs maximum pooling. The output of max pooling compresses the n points in the input point cloud to a subset of points, similar to how the max operator compresses numerous real-valued inputs to a single value. In reality, the global feature vector can be contributed to by no more than K points. The critical point set is made up of points that contribute to and define the Global Feature Vector, and it encodes the input with a sparse collection of key points.

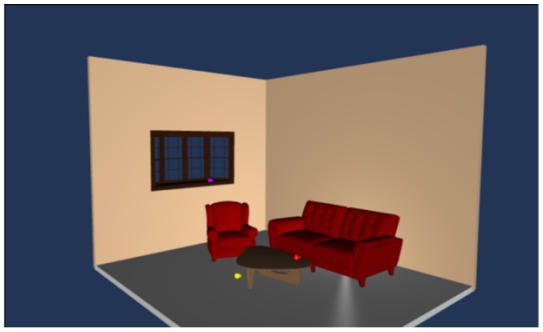
More intriguingly, the network learns to summarize an input point cloud by using a sparse collection of important points, which closely matches to the skeleton of objects according to visualization. The original shape of an object is represented as a dense cloud of points and the critical points are created in the areas where the highest density is found, therefore the skeleton of the object.

2.1 Training and Deploying machine learning model into a website

To display a machine learning model in a website you need to decide whether to use a trained model or train it into a website. In our case we imported a pre-trained model.

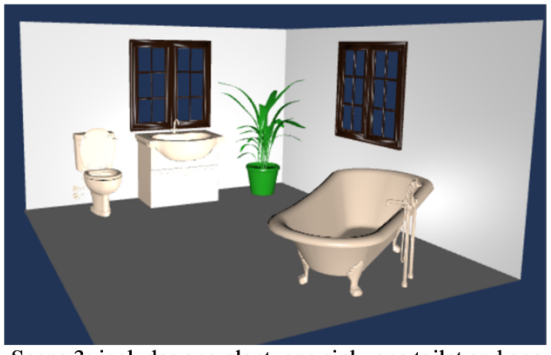
For the needs of training we use ModelNet40, a large dataset with 40 classes of 3D objects. By following the PointNet architecture and its implementation with Google TensorFlow in Keras framework, we built the model to handle point clouds with a constant number of points. Then we set a batch size equal to 32 and train the neural network for 20 epochs. This means that the whole dataset is organized in groups of 32 clouds and each of them goes through our network exactly 20 times.

For running the Deep Learning algorithm in the webpage we use a converter implemented in Tensorflow.js, an open-source toolkit that uses Javascript and a high-level layers API to create, train, and run machine learning models fully in the web. So we save our trained neural network ("model") in ".json" format.



Scene 1: includes a sofa, an armchair, a table and 4 cups.

Scene 2: includes a bed, a stand, a chair, a desk, a laptop and a plant.



Scene 3: includes one plant, one sink, one toilet and one bathtub.

Scene 4: includes one piano, a lamp, one piano chair, a curtain, one bookshelf and two deferent guitars.

Figure 3: The four individual scenes we use to evaluate the performance of the proposed methodology. The examples are available also in https://www.medialab.hmu.gr/minipages/3DRtree/

3 REAL TIME GEOMETRY CONDITIONING

3.1 Resampling

Our trained model takes a constant number of points as input and outputs the predicted result. Thus all the objects we wish to predict by our algorithm need to have the same number of points with the ones we used for training. However, web3D models are expected to have different and random numbers of points. Thus, in order to predict such objects we need to equalize the number of points of each one of them by increasing or decreasing their point clouds where it is necessary. To achieve this task we first set a "target" value that equals the final number of points we wish for each object. Then we raise three cases.

- In the first case the object has a greater number of points than the "target" so we down sample the points until the length equals "target".
- In the second case the number of object points are less than the "target". In that case we oversample each point cloud.
- Finally, if the number of points equals to "target" value we save the point cloud as it is.

On the other hand PointNet predicts better when the point clouds are rather homogenously distributed in the volume (similarly to laser scanner point clouds). However, the point clouds we can get from ".x3d" objects are sparse because they are intensively "light" in number of points since they are mostly for the web. Sparsity is a factor that can confuse PointNet and lead the prediction to the wrong result due to the inability of exporting critical (skeleton) points. For this reason we over sample points in ".x3d" models by interpolating points inside faces, triangles or squares that form the surface of an object, by using data from IndexedFaceSet. Mainly we are using a modification of the Wolfram Mathworld Triangle Point Picking [Wolfram 2022]

$$x = a_1 v_1 + a_2 v_2 \tag{1}$$

, wherev_1,v_2 are two vertices of a triangle, the third vertex is at the origin, and a_1,a_2 are uniform varieties in the interval [0 , 1]. In our case, we found the minimum and the maximum value of each of the tree axis and we add points to this bounds, then toachieve a better distribution of points, we fill the areas with points in random

positions and in density that is related to the area of each face.

$$triangle\ area = \frac{1}{2}\left[x_1\left(y_2 - y_3\right) + x_2\left(y_3 - y_1\right) + x_3\left(y_1 - y_2\right)\right] \quad (2)$$

Thus with (2) we add more points to bigger areas and less points to smaller areas.

4 EXPERIMENTAL RESULTS

We evaluate the performance of PointNet in ".x3d" object by doing two different experiments. In the first experiment (Case 1) we evaluate efficiency of the algorithm in classification prediction of several types of 3D model from the ModelNet40 dataset that has been transformed to X3D format. In the second test (Case 2) we created four X3D scenes with different objects and we evaluate the classification results.

Training of the Deep learning algorithm was executed with ModelNet40 models in different number of points and different epochs. In Table 1 there are the two different configurations of the dataset and the corresponding training accuracy. The Accuracy is rather lower than expected because we feed the training set with all the ModelNet40 (40 classes, ~800 models in each class for training) and all these in two different point densities (3000 points and 6144 points). The lager the number of the training classes the lower the expected accuracy.

The execution of prediction experiments are running in webpage environment in order to evaluate our methodology but also to evaluate the feasibility of running the algorithm in the web browser environment.

4.1 Case 1: Evaluation of the algorithm in the classification of individual X3D models

In Table 2 we present the results of the experiments we made with the algorithm that has been trained for different kind of point densities and for 20 epochs which shows better training accuracy. All the models are in X3D format.

The experiments has been applied to several classes of 3D models (chairs, sofa, table, bathtub, bed, book, guitar, lamp, piano, plant, sink, stand). For each class we pick 10 3D models from the ModelNet40 testing dataset (different than training dataset we used for training) and 10 models randomly from the internet. The results in Table 2 reflect the percentage of positive predictions, Average Precision in detection of the correct class during the tests. The classification prediction efficiency appears to be over 80% in most of the cases. Of course classification performance is related to geometric complexity of the model we are testing. For example a piano or a plant is more complicated geometry of a sofa or a table. Also it is interesting to notice that performance is higher in the classification of low point models and decreases as the number of points increases.

It is obvious that the closer the number of points of the tested models to the number of points that we trained the Deep-Learning algorithm the better the performance.

The corresponding Mean Average Precision in case of MeshNet algorithm in [Feng et al., 2018] is 81.9, however anyway, the performance as we already mention has to do with complexity of the models and the possibilities of classes the algorithm has to predict.

Table 1: Training Configuration

Points	Epoch	Accuracy
3000	10	76.35%
3000	20	78.26%
6144	10	73.96%
6144	20	75.04%

Table 2: Classification performance (Average Precision(%)) (*MAP=Mean Average Precision)

Points	chairs	Sofa	Table	bathtub	Bed	Book
3000	80	85	90	85	90	80
6144	85	95	95	100	95	90
9000	85	80	90	95	90	90

Points	Guitar	Piano	Plant	Sink	Stand	MAP*
3000	80	65	65	80	85	80.45
6144	90	65	65	80	85	85,9
9000	90	75	75	80	85	85

4.2 Case 2: Evaluation of the algorithm to complex X3D scenes with different models in the same scene

In the second case we created four scenes with different models appear in each one of them (Figure 3).

In Table 3 we present the results of the evaluation. We may notice that scenes with models that have been predicted accurately in Table 2 have higher rate of prediction in Table 3 as well.

The results of the above tables are showing us that:

a) Typical X3D models are not dense enough in order to achieve high prediction accuracy during classification process, thus resampling is required.
b) The closer is the number of points of the testing models (enhanced X3D models) to the number of points of the training models the better is the accuracy we achieve. Here the training set contains models with 6144 points. So as we can see by the results, in general the better efficiency is achieved when testing models has number of points closer to 6144 total points.
c) In some of the above cases objects were correlated with classes that do not match their appearance. PointNet algorithm is not using surface or appearance characteristics in order to classify models. Instead it is using Global Feature Vector which is extracted by Max-Pooling to a transformed version of the point set.

5 CONCLUSIONS

In this paper we are studying the capability of X3D models to work in a deep learning environment. We are using a point set classification algorithm because it is more appropriate for dynamically created models from sensor scanning or photogrammetry. We choose X3D models because they are semantic friendly which

Table 3: Complex X3D scenes objects prediction (Figure 3)

Points	Scene 1	Scene 2	Scene 3	Scene 4
3000	5 correct in 7 objects	3 correct in 6 objects	3 correct in 4 objects	4 correct in 7 objects
6144	5 correct in 7 objects	4 correct in 6 objects	3 correct in 4 objects	5 correct in 7 objects
9000	4 correct in 7 objects	4 correct in 6 objects	3 correct in 4 objects	4 correct in 7 objects

is a feature valuable in the modern information society. In our results we show that X3D models enhanced with some resampling of the vertices, especially in models with large polygons, are capable enough to work with deep learning algorithms. So the major contribution of this work is that we prove that PointNet algorithm can be used for meshes as well as for point clouds having similar performance to one of the latest meshes classifications algorithm. Even more we show that even sparse meshes like those in Web3D can be classified as well. This conclusion creates new opportunities in the use of X3D as a modeling language of real time scanning created and annotated scenes. In our future work we will focus on segmentation of point sets and the adoption of X3D format as a point set declaration format.

REFERENCES

Y. Guo, H. Wang, Q. Hu, H. Liu, L. M. Bennamoun. 2021. "Deep Learning for 3D Point Clouds: A Survey," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 12, pp. 4338-4364, 1 Dec. 2021, doi: 10.1109/TPAMI.2020.3005434

Ha-Seong Kim, Myeong Won Lee. 2020. "3D Object Recognition Using X3D and Deep Learning", Web3D '20: The 25th International Conference on 3D Web Technology, November 2020, Article No.: 10, Pages 1–8, https://doi.org/10.1145/3424616.3424703

Charles R. Qi, Hao Su, Kaichun Mo, Leonidas J. Guibas. 2020. "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation", DOI: 10.48550/arXiv.1612.00593

J. Flotyński, A.G. Malamos, D. Brutzman, F.G. Hamza-Lup, N.F. Polys, L.F. Sikos. 2020. Recent Advances in Web3D Semantic Modeling in Book Recent Advances in 3D Imaging, Modeling, and Reconstruction, pp23-49, IGI Global Publications, 2020

J. Flotynski, D. Brutzman, F. G. Hamza-Lup, A. G. Malamos, N. Polys, L. F. Sikos, K. Walczak. 2019. "The Semantic Web3D: Towards Comprehensive Representation of 3D Content on the Semantic Web", International Conference on 3D Immersion (IC3D), Brussels, Belgium, 2019

Mathworld Wolfram. 2022, https://mathworld.wolfram.com/TrianglePointPicking.html

MODELNET PRINCETON. 2022, https://modelnet.cs.princeton.edu/#

W. Zhou, J. Jia, C. Huang, Y. Cheng. 2020. "Web3D learning framework for 3D shape retrieval based on hybrid convolutional neural networks," in Tsinghua Science and Technology, vol. 25, no. 1, pp. 93-102, Feb. 2020, doi: 10.26599/TST.2018.9010113

W. Zhou, J. Jia, W. Jiang, C. Huang. 2021."Sketch Augmentation-Driven Shape Retrieval Learning Framework Based on Convolutional Neural Networks," in *IEEE Transactions on Visualization and Computer Graphics*, vol. 27, no. 8, pp. 3558-3570, 1 Aug. 2021, doi: 10.1109/TVCG.2020.2975504

Yutong Feng, Yifan Feng, Haoxuan You, Xibin Zhao, Yue Gao. 2018. "MeshNet: Mesh Neural Network for 3D Shape Representation, " in https://doi.org/10.48550/arXiv.1811.11424