Meshing Point Clouds with Predicted Intrinsic-Extrinsic Ratio Guidance

Name:	**Vimaleswar A**
Registration No./Roll No.:	20305
Institute/University Name:	IISER Bhopal
Program/Stream:	EECS

1 Abstract

Our research project focuses on reconstructing mesh representations from low-resolution point clouds, obtained from sensors like LiDAR, Kinect etc. Current learning-based mesh generative methods struggle with fine-grained details and generalization to new object categories due to their reliance on shape embeddings at the whole-object level. In contrast, the method proposed in this paper maximizes the utilization of the input point cloud by incorporating connectivity information directly onto existing points, predicting which triplets of points should form faces. They introduce a novel surrogate of local connectivity, computed through intrinsic/extrinsic metrics and learned using a deep point cloud network.

2 Introduction

Reconstructing high-quality 3D meshes from point clouds is crucial for various real-world applications like autonomous driving and robotics. Traditional methods either connect points directly or approximate surfaces, but struggle with ambiguous structures (like thin structures, corners) and often require manual parameter tuning. With the rise of 3D deep learning and large-scale datasets, people tend to learn geometric priors from data. However, generating meshes directly from neural networks is challenging due to the irregular nature of meshes.

The method proposed in this paper addresses this by leveraging triangle faces as an intermediate representation. This method excels in preserving fine details and handling ambiguous structures with learned intrinsic-extrinsic guidance. As, it mainly relies on local geometric information, it generalizes well to unseen categories. Experimental results on ShapeNet dataset show superior performance compared to traditional and learning-based methods across various metrics which includes the F-score, Chamfer distance and normal consistency. Extensive ablation studies demonstrate this method's generalizability and robustness.

3 Related Work

3.1 Traditional Mesh Reconstruction

Traditional mesh reconstruction methods mainly include two paradigms:

- **Explicit reconstruction:** Ball-pivoting algorithm(BPA), Delaunay triangulation, alpha shapes and zippering. These methods resort to the local surface connectivity estimation and connect the sampled points directly by triangles.
- **Implicit reconstruction:** Poisson surface reconstruction(PSR). These methods aim to approximate the point cloud with a field function, such as a signed distance function. They then use the marching cubes algorithm to extract the surface of this field function.

3.2 Learning-based mesh generation

Existing learning-based mesh generative methods mainly follow five paradigms:

- **Deformation-based methods:** It starts with a basic mesh shape, (like sphere) and then deform it into the desired shape. As they only change vertex positions without altering connectivity, they may struggle to create shapes with specific topologies.
- **Folding-based methods:** They learn a set of mappings from 2D squares to 3D patches, which are then used to form mesh.
- **Primitive-based methods:** It uses simple shapes, (like planes or convex patches) as building blocks for the mesh. They learn the parameters of these shapes. However, these simplicity might limit their ability to capture fine details.
- **Optimization-based methods:** They either use deep neural networks as local geometric priors or learn local shape priors from data. They treat mesh reconstruction as an optimization problem and so computationally expensive.
- **Implicit field-function-based methods:** This uses neural networks to learn an implicit field function that represents the shape. After they apply marching cube algorithms to extract the surface.

4 Method

Given a 3D point cloud P = (x_i, y_i, z_i), we aim to reconstruct a polygon mesh, which consists of a vertex set and a face set, approximating the underlying surface.

4.1 Remeshing Algorithm

Given a reference triangle mesh (M_R) and a point cloud P on sampled on M_R as input, this algorithm generate a new mesh M_N whose verices come from the point cloud P. In a broad view, this method leverages ground point clouds and uses the IER (Intrinsic-extrinsic ratio) as training guidance to differentiate between surface triangles from non-surface triangles i.e., it utilizes the ratio of geodesic distance and euclidean distance as guidance to train the network for classifying candidate triangles.

The detailed process is as follows: First, the algorithm construct a k-NN graph for each points in P and then, it construct a candidate triangles. Since, reference mesh M_R is given, we can calculate the geodesic distance between two vertices. Next, we calculate the intrinsic-extrinsic ratio (IER) for each candidate. We filter out the incorrect candidate triangle with $IER \geq \tau$ and $\tau \geq 1$. Then, sort the remaining candidates based on their distance to M_R and the length of their longest edges. After visiting all the candidates, we get the final mesh M_N.

4.2 From Remeshing to Reconstruction

Unlike the Remeshing Algorithm. Here, we are given only a point cloud P as input and our aim is to reconstruct the mesh M_N. Since, reference mesh M_R is not given, we cannot directly calculate the intrinsic-extrinsic ratio. So, neural network will be helpful for estimating the local connectivity and geometry of input point cloud. We used neural network to filter out the incorrect candidate triangles and provide cues for sorting the remaining candidates. Next, divided the candidates into l+1 categories and in this paper they used three types of class labels 0,1,2 . Here, 0 means which are incorrect candidates(IER ¿ π) triangle, 2 means which are near to ground truth surface and 1 means which are very close to ground truth surface.

5 Experiments

Various experiments are conducted to see how well this method works for turning point clouds into 3D models. We sampled 23,108 synthetic CAD models, which cover eight categories, from the ShapeNet dataset to evaluate our model. Out of these 23,108 synthetic CAD models, we reserved 3,146 models for testing and rest for training. Then, we train the model on a powerful computer for many rounds, with different settings like we constructed a k-NN graph where the value of k is 50. Then, we used the MMP algorithm to calculate the exact geodesic distance between each vertex and its small neighborhood over the mesh surface. We then set the τ value to 1.3 to filter out the incorrect candidates. Then, the correct candidates are divided into two categories according to their distances to the mesh surface with a threshold value of 0.005. On comparing the method with other methods commonly used for this task, like traditional methods and newer ones based on machine learning. It well preserved the details and handled tricky parts of the models. After running lots of tests and crunching numbers, it is found that this method performed better than the others in almost every aspect. It could capture fine details and handle complex shapes with ease. Try different scenarios to make sure this method could handle different kinds of data and noise levels, and it perfoms good. It can be concluded that this new approach is excellent for turning point clouds into detailed 3D models, making it a valuable tool for various applications.

6 Steps to run the code

First, we connect to IISERB Group17 server which is given to us. Then, we clone the github repository where the code is present in the Visual studio. Then, we download the datasets. Then, we made virtual environment and download the prerequisite libraries(like tqdm, pickle5, numpy etc). Then installed the Python 3.8 and torch 2.2.1 version. Then we followed the readme file instruction . But during the running develope.sh we faced the problem that the coda version is not matching with the installed torch version so again we changed the torch version to torch version 1.13. Then, again we updated the torch version to 2.2.1. Next, we run the test.py which produced the test demo data(labels data) using pretrained model. Then we used this test demo datasets to postprocess models which are given C++ file so as per mentioned in Readme file we downloaded and uploaded the annoy 1.16 version. Then we run all the postprocess C++ file. Refer Figure 1. Next, we run main.py file and rundemo.py file which produce the output mesh points (x,y,z coordinates) refer Figure 2. Then using online 3D viewer we converted the output points to the respective 3D images.

7 Output

The below shown are the Figures 3, 4, 5, 6, 7, 8, 9 which we got as output after doing the postprocessing in ShapeNet test set for the given point clouds. Figure 10 shows the zoomed in figure of Aeroplane where we can see the filtered candidate triangle merged together to form the Aeroplane object. We now train the model by using train.py and train datasets. We took the parameters like batch size as 30, decay rate as 0.0001, epoch is 1 (As it takes more time if the number of epochs increases), gpu = 2(for using GPU). We got training accuracy scores, validation loss scores. Refer Figure 11.

8 Conclusion

In this paper, they proposed a novel learning-based framework for mesh reconstruction that is based on the grounded point clouds and explicitly estimates the local connectivity of the points. By leveraging the intrinsic-extrinsic ratio as training guidance, the method is able to effectively distinguish the surface triangles and non-surface triangles. Extensive output have shown our superior performance, especially for preserving the details, handling ambiguous structures, and strong generalizability.

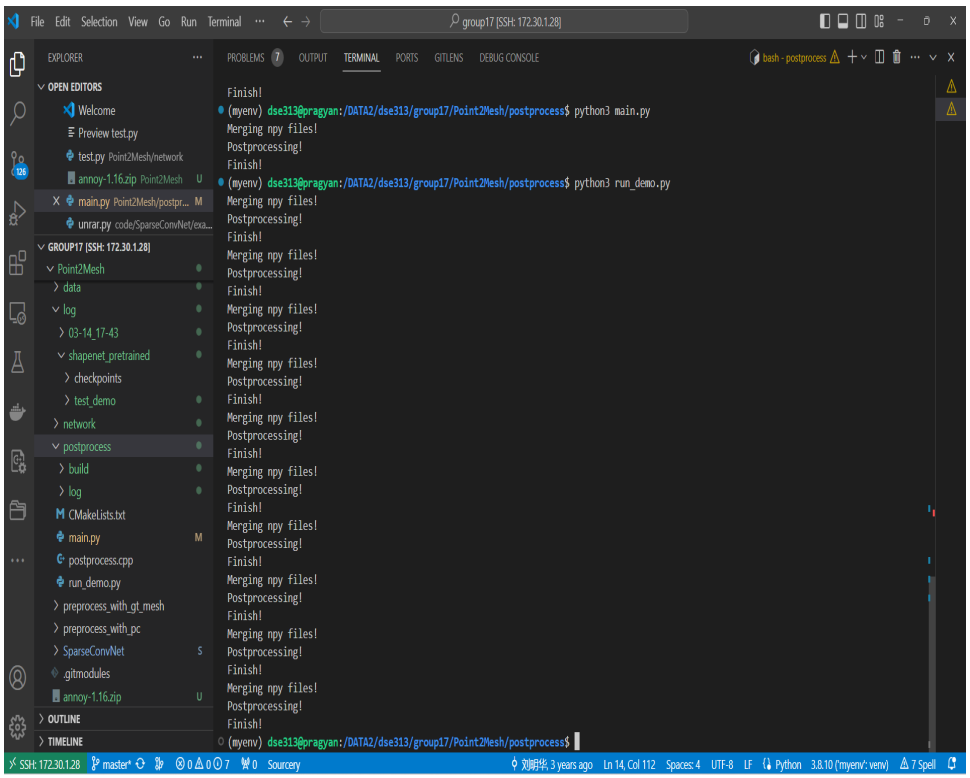

Figure 1: Susscessfully completed Postprocessing

9 Future Work

The below is the possible novel approach for further progress(which is not limited to/or the below idea may be any changed based on the time availability and output). We can use GNN(Graph Neural Network). We have an idea that if we use the triangles points as nodes and relation between two points as edge then we will convert the mesh points into GNN data. So, we can use GNN models to get better result.

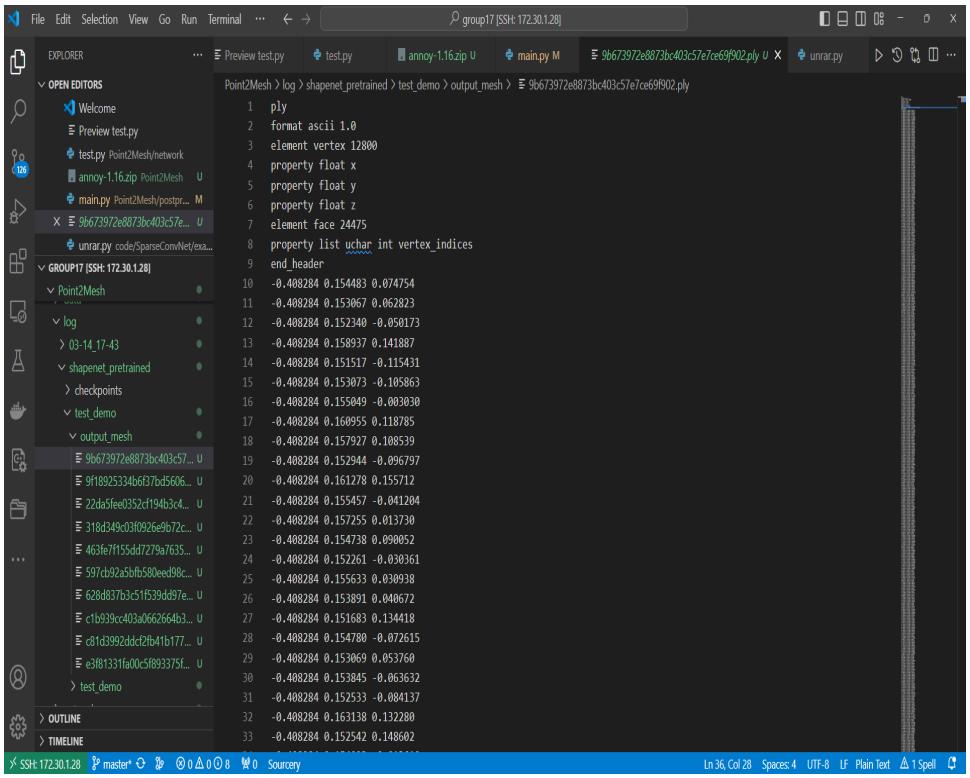

Figure 2: Output points after postprocessing

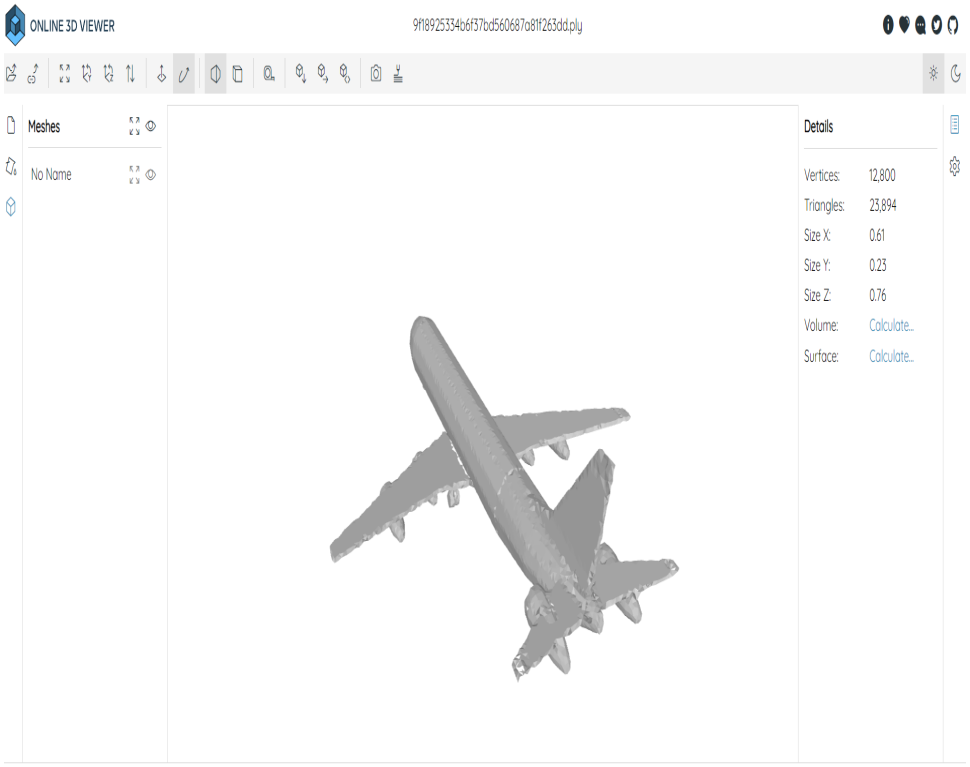

Figure 3: Reconstructed meshes of the ShapeNet test set of Aeroplane

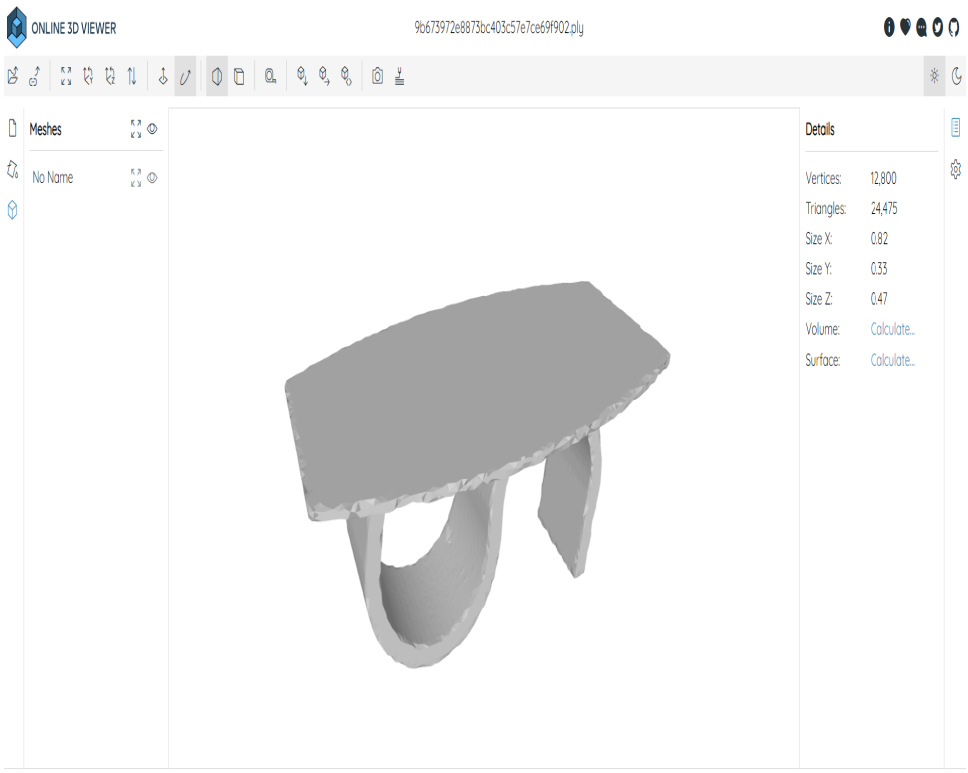

Figure 4: Reconstructed meshes of the ShapeNet test set of Table

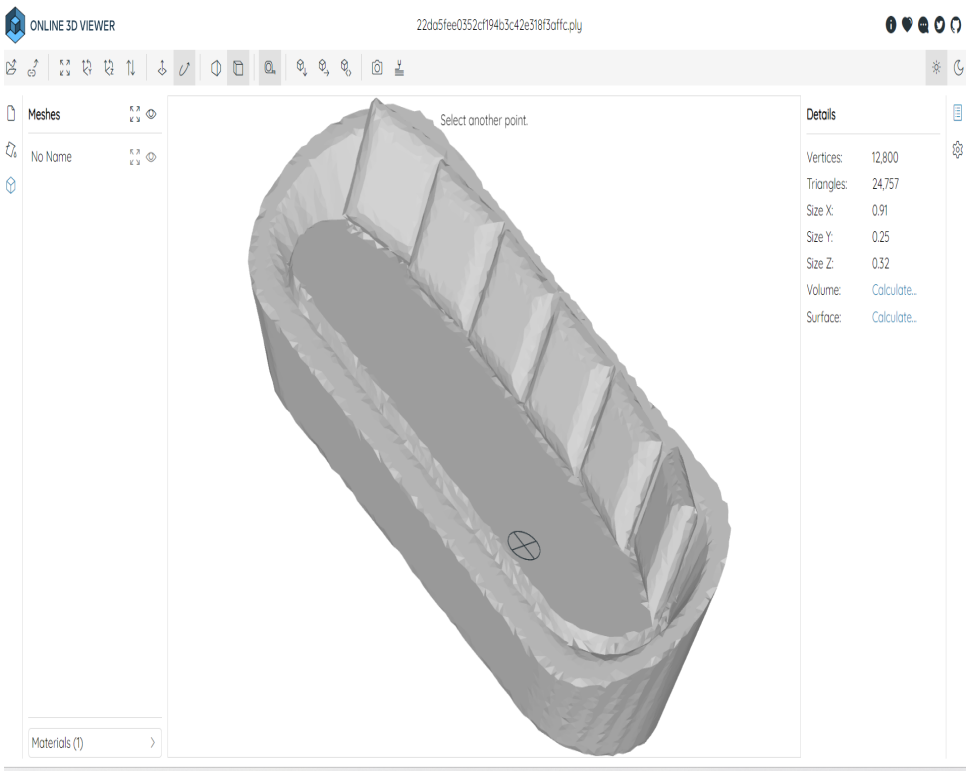

Figure 5: Reconstructed meshes of the ShapeNet test set of Sofa

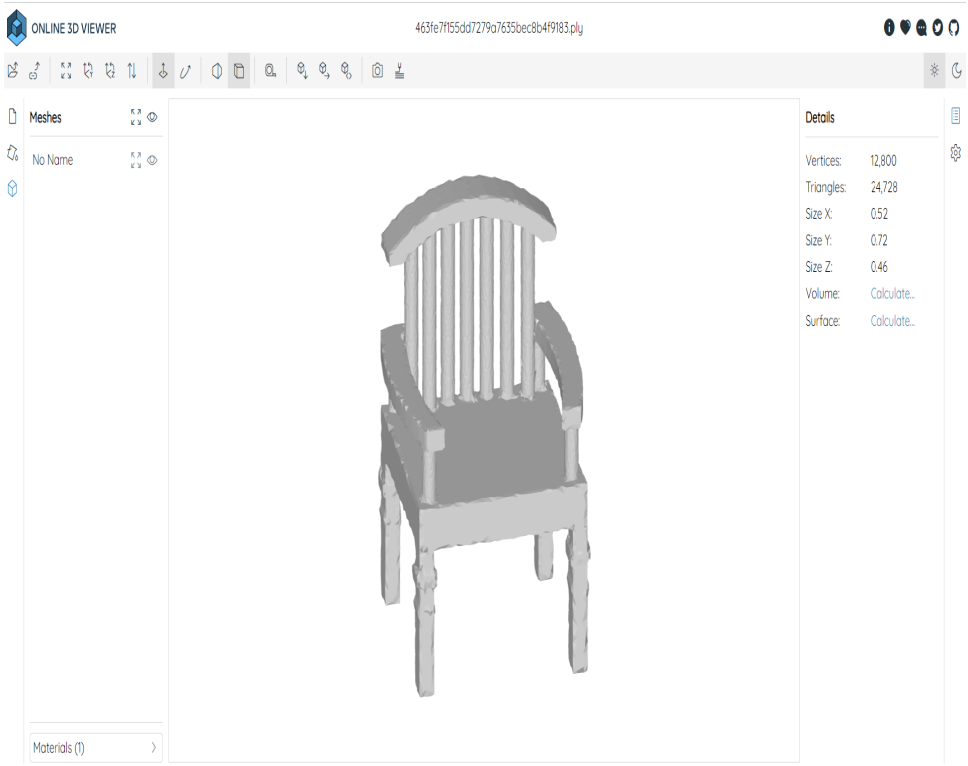

Figure 6: Reconstructed meshes of the ShapeNet test set of Chair

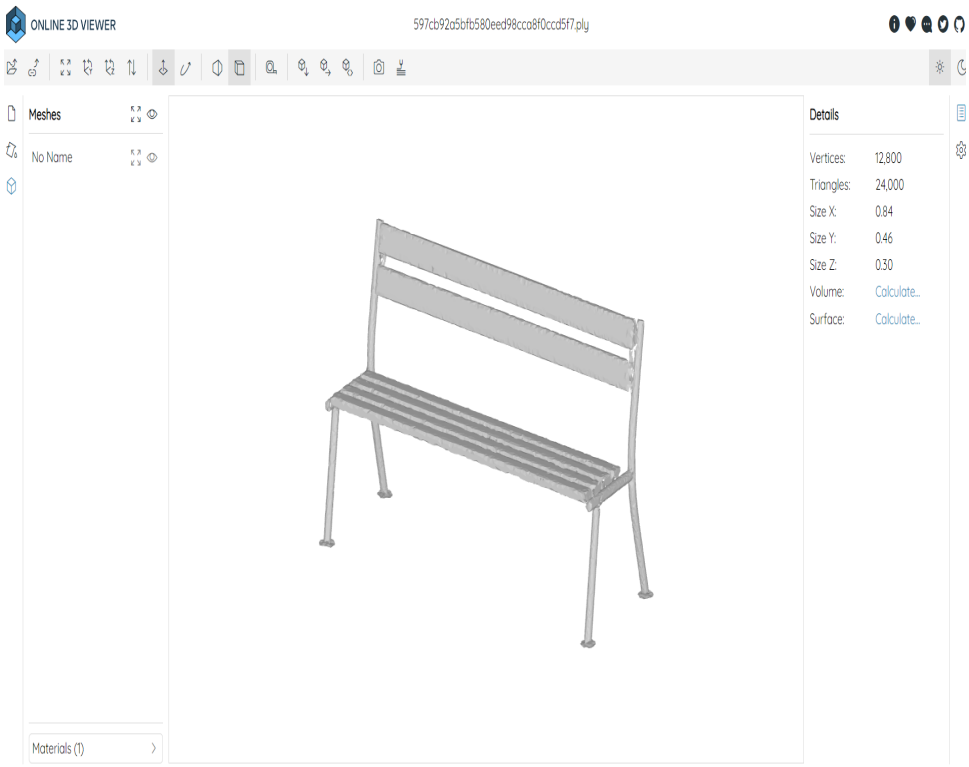

Figure 7: Reconstructed meshes of the ShapeNet test set of Bench

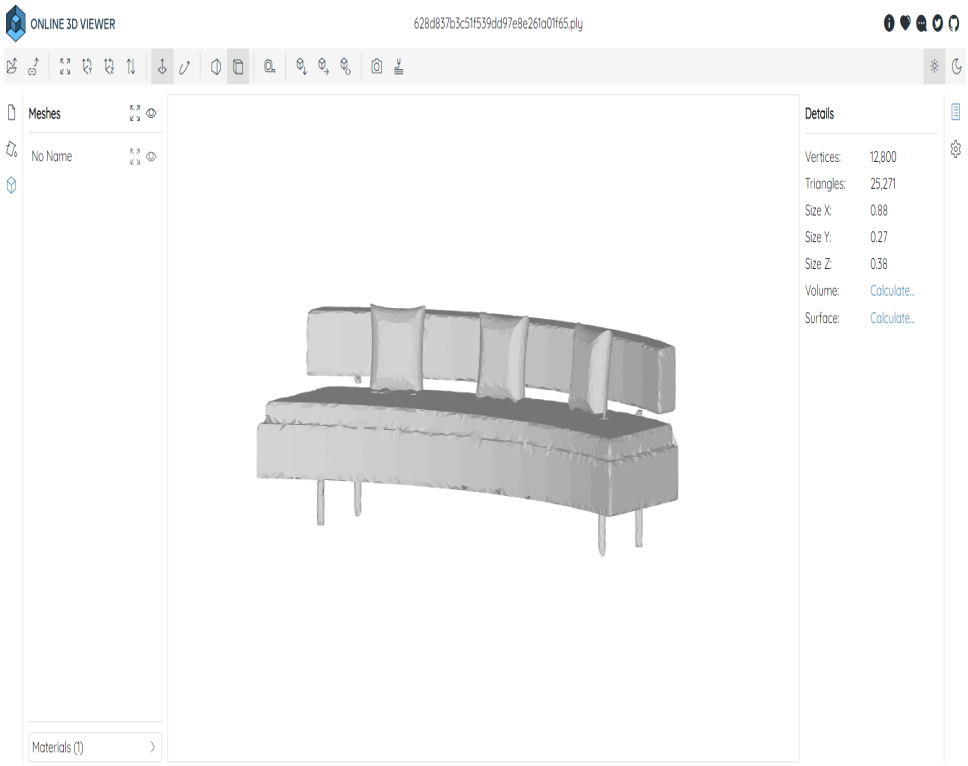

Figure 8: Reconstructed meshes of the ShapeNet test set Chair Sofa

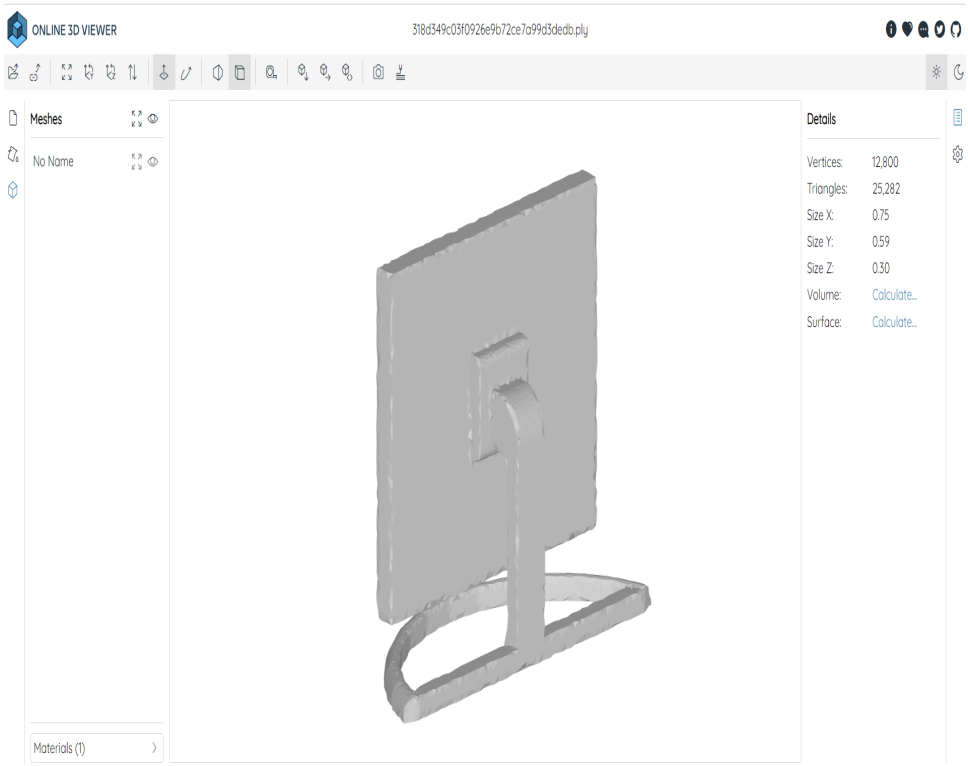

Figure 9: Reconstructed meshes of the ShapeNet test set Desktop

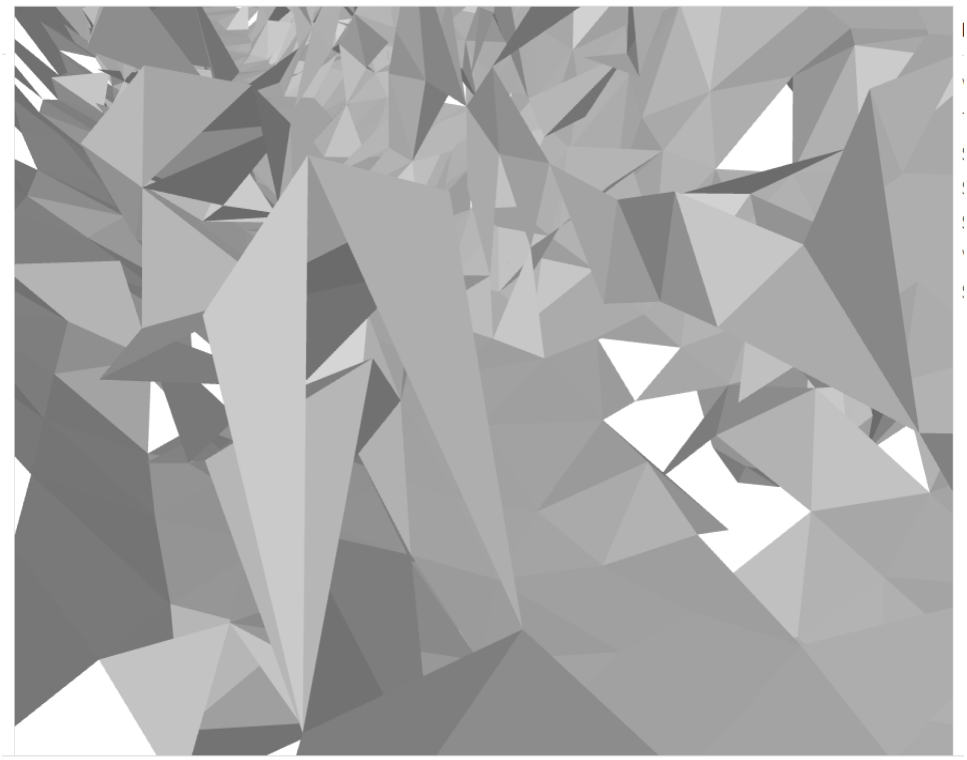

Figure 10: Filtered candidate triangle which merged to form an Aeroplane

```
2024-04-18 20:57:08,110 - INFO - PARAMETER ...
2024-04-18 20:57:08,110 - INFO - Namespace(batch_size=30, dataset_dir='../data/all/all/', decay_rate=0.0001, epoch=1, gr
2024-04-18 21:13:45,757 - INFO - PARAMETER ...
2024-04-18 21:13:45,757 - INFO - Namespace(batch_size=30, dataset_dir='../data/all/all/', decay_rate=0.0001, epoch=1, gr
2024-04-18 21:13:45,757 - INFO - Train data loading 1
2024-04-18 21:20:39,759 - INFO - Train data loading 2
2024-04-18 21:20:39,759 - INFO - Val data loading 1
2024-04-18 21:22:59,710 - INFO - Val data loading 2
2024-04-18 21:22:59,711 - INFO - The number of train data is: 19962
2024-04-18 21:22:59,711 - INFO - The number of val data is: 3146
2024-04-18 21:23:13,405 - INFO - No existing model, starting training from scratch...
2024-04-18 21:23:13,406 - INFO - Epoch 1 (1/1):
2024-04-18 21:23:13,406 - INFO - Learning rate:0.001000
2024-04-18 21:23:13,406 - INFO - BN momentum updated to: 0.100000
2024-04-18 22:11:42,625 - INFO - Train accuracy: 0.78952
2024-04-18 22:11:42,626 - INFO - Train loss: 0.89891
2024-04-18 22:11:42,626 - INFO - Train loss_2: 0.39462
2024-04-18 22:11:42,626 - INFO - Train loss_3: 0.50429
2024-04-18 23:01:37,479 - INFO - Val accuracy: 0.84154
2024-04-18 23:01:37,480 - INFO - Val loss: 0.73733
2024-04-18 23:01:37,480 - INFO - Val loss_2: 0.32516
2024-04-18 23:01:37,480 - INFO - Val loss_3: 0.41217
2024-04-18 23:01:37,480 - INFO - Saving model...
2024-04-18 23:04:37,747 - INFO - Saving model...
2024-04-18 23:05:37,846 - INFO - Best accuracy is: 0.84154
```

Figure 11: Train result after 1 epoch run