CMPT 764 Final Project

Atmika Honnalgere: 301408973
Janaki Sathish: 301396056
Priyanka Chandrasekhar: 301393758
Samuel Hatin: 301394799

Contribution

- Atmika Honnalgere: Part mixing, part resizing and replacement, PointNet
- Janaki Sathish: Scorer: ResNet, Comparison between ResNet vs LeNet vs PointNet
- Priyanka Chandrasekar: Point cloud generation, part warping, part movement and final set view generation for point clouds.
- Samuel Hatin: Parsing code for PartNet, IO, dataset generation for classifier training and testing . Training and testing of Extended LeNet

Objective

- Design and implementation of an automated part assembly 3D modelling method to generate 3D models of chairs
- Building a scorer that can score the plausibility of the generated chairs

Parser

The parser runs on the PartNet-Symh dataset obb files to create a dataset of *Mesh* objects from the set of models provided to the parser.

Each *Mesh* object contains a list of *Part* objects. Each *Part* object corresponds to a physical part of the object (found in the .obj file).

OBB files

Extracting the required information like bounding boxes, connectivity and labels was done by reading the .obb files provided in the PartNet-Symh dataset.

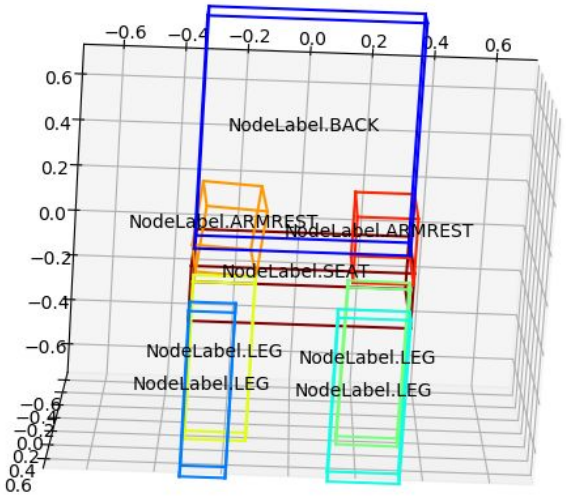
```
1  N 3
2  0.0302842 0.488309 -0.315397 -0.995024 0.0971271 -0.0222212 -0.00601635 0.164046 0.986434 0.0994548 0.981659 -0.162646 0.89214 0.305279 0.781198
3  0.00419755 -0.422335 0.169623 0.999916 0.0121707 -0.00438424 -0.0121707 0.770206 -0.637679 -0.00438424 0.637679 0.77029 0.898086 0.523685 1.15351
4  0.0030865 0.148677 0.0724135 1 0 0 0 1 0 0 0 1 0.890567 0.223845 0.805681
5  C 2
6  0 2
7  1 2
8  S 0
9  L 3
10 0
11 2
12 1
```

Parser

Each *Part* object has the following attributes:

- **vertices**: List of (x,y,z) vertex coordinates corresponding to the part
- **faces**: List of faces stored as vertex indices corresponding to the faces of the part.
- **label**: BACK=0, SEAT=1, LEG=2, ARMREST=3
- **bounding_box**: The 8 coordinates for the corner points of the oriented bounding box.
- **render**: A flag to determine if the part is to be rendered.
- **color**: (r, g, b) tuple to apply color to the part mesh while rendering

Display Bounding Boxes with Labels



Mixer

The mixer chooses a *target* Mesh object at random from the dataset created by the parser.

Each chair part in the target is to be replaced by corresponding chair parts sourced from other chairs (chosen at random) in the dataset. The target mesh only provides the base structure on which the replacement and deformation is based.

Each chair part in the target is either -

- Replaced by parts deformed to match the shape of the target parts
- Directly replaced by the source parts
- A transformed target part, in case no match is found in the dataset (example an armrest)

Part deformation

The mixer uses non-rigid point set registration to deform the source parts to the target mesh parts.

The deformation is applied only when the number of target and source parts are equal. E.g. if the target model has 4 legs, but the source has only 1, warping is not applied.

Non-Rigid Point Set Registration

Algorithm : Coherent Point Drift

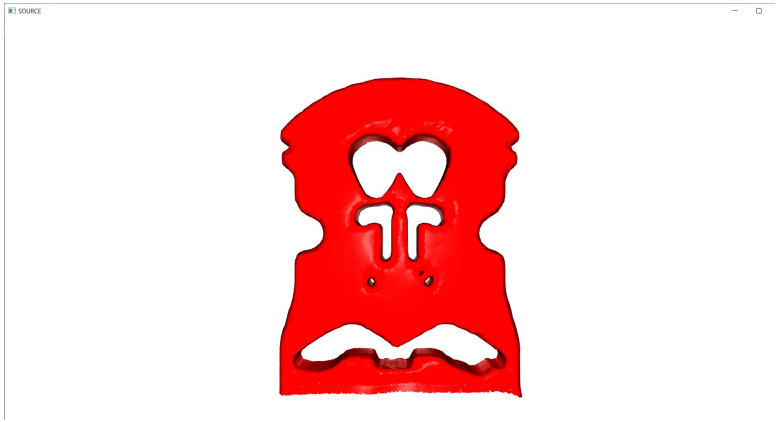
CPD is a probabilistic method of assigning correspondences between two sets of points and recovering the transformation matrix that maps one set to another. CPD can recover non-rigid transformations that can be used to warp the source part into a shape similar to the target part.

Implementation

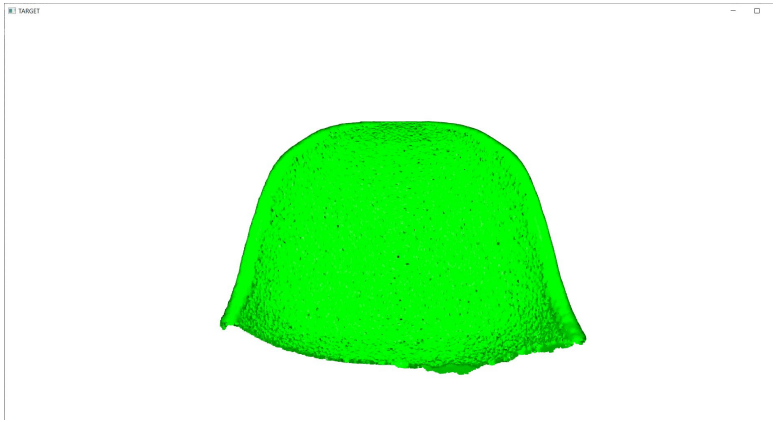
To implement CPD we used the library Probreg (probablistic point cloud registration library), which uses Open3D as an interface and implements various kinds of point cloud registration algorithms, of both rigid and non rigid kind. Since this library works well with point clouds, we have created point clouds from the input triangle mesh models for the deformation and further matching of parts.

Source

Part deformation results

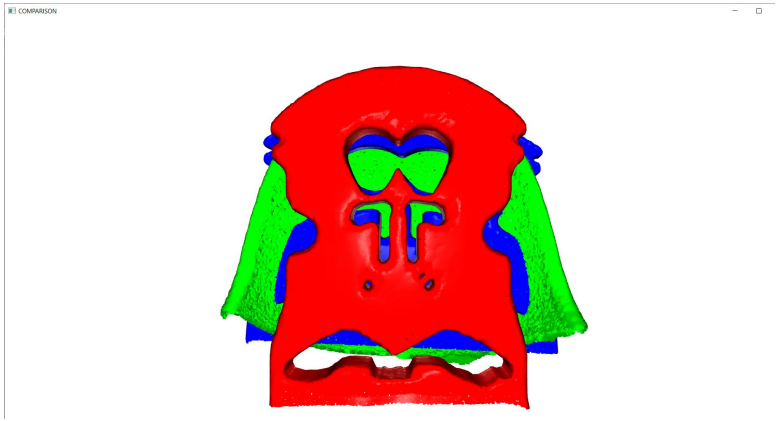


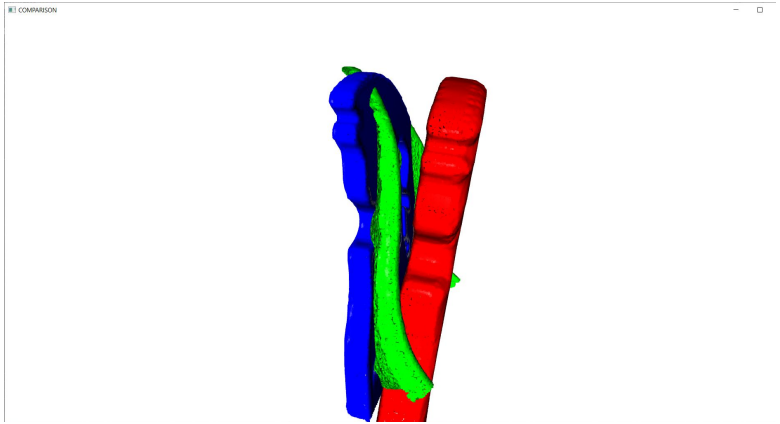
Source part



Target part

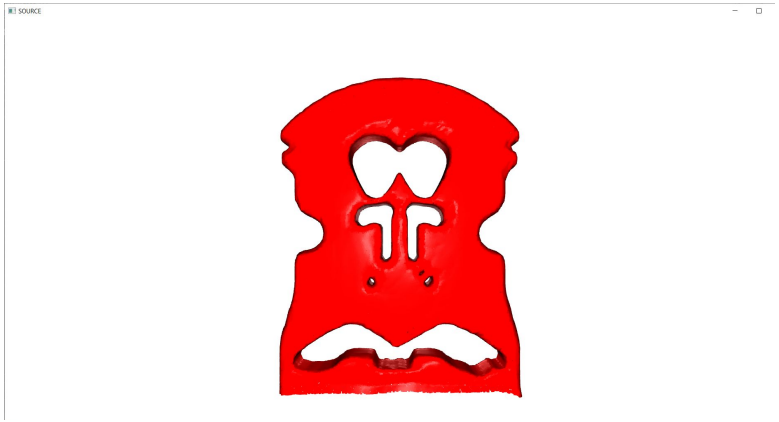
Part deformation results



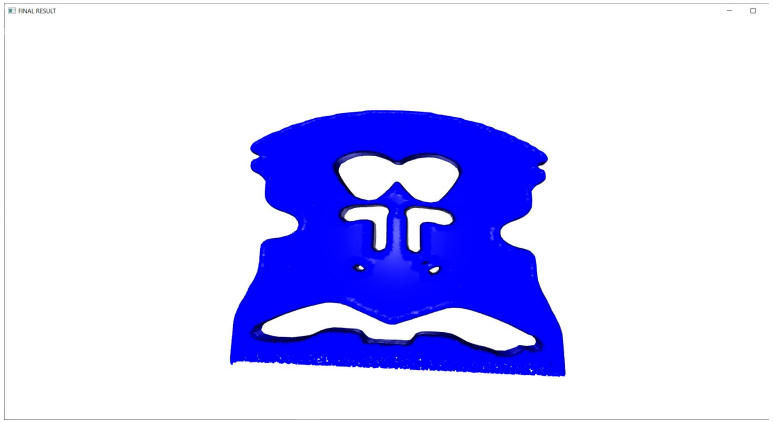


Comparison between parts. Here red is the source, green is the target and blue is the result of deformation. Compared to the source the result is thinner, tilted and translated too like the target.

Part deformation results



Source part



Result part (after warping source)

Resizing/Replacing parts

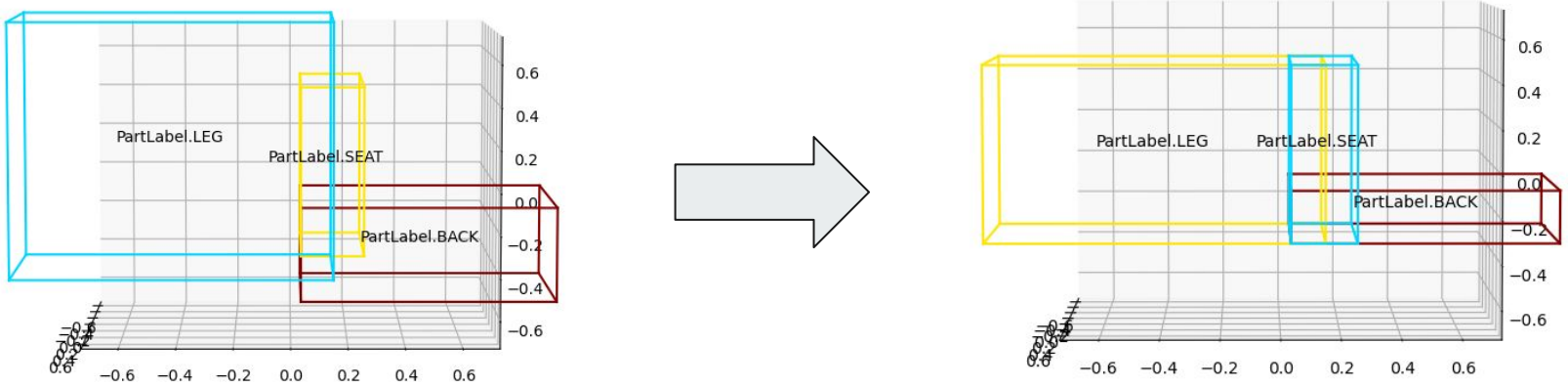
Source parts, both warped and un-warped, need to be resized and translated so that they align with the overall chair structure of the target.

We compute a bounding box that encloses all parts with the same label, e.g., all 4 legs for both the source and target parts and use their positions and dimensions to compute the affine transformation for the source mesh vertices.

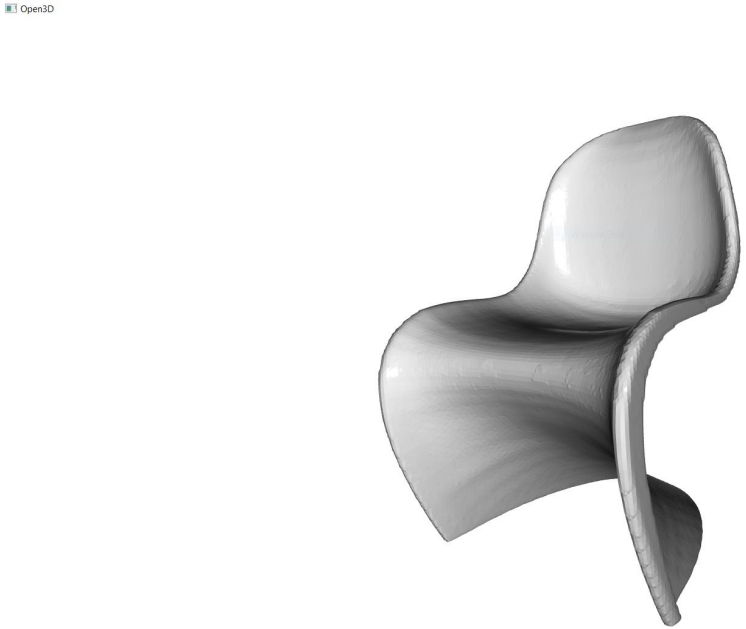
Resizing/Replacing parts

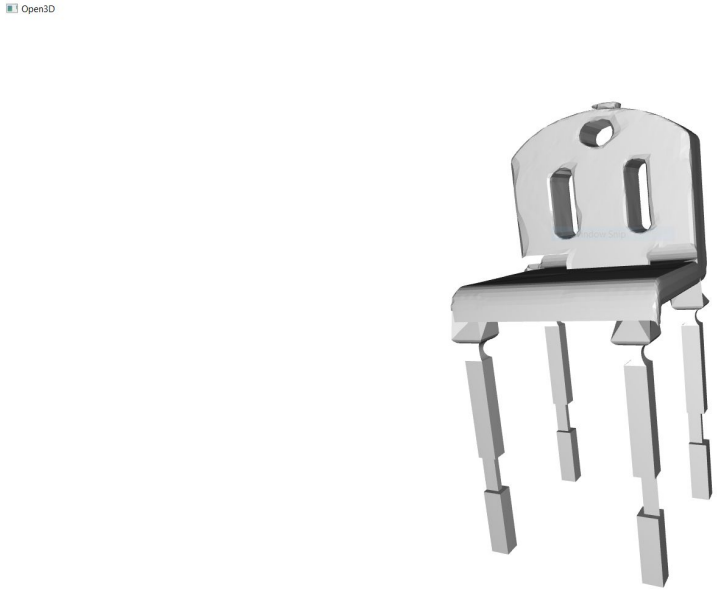
For LEG, BACK and ARMREST

To handle cases where the source parts may be misaligned with the structure, we resize the target bounding box for these labels to match the bounding box of the SEAT in the x-z plane. The dimensions in the y-direction (height) are maintained.

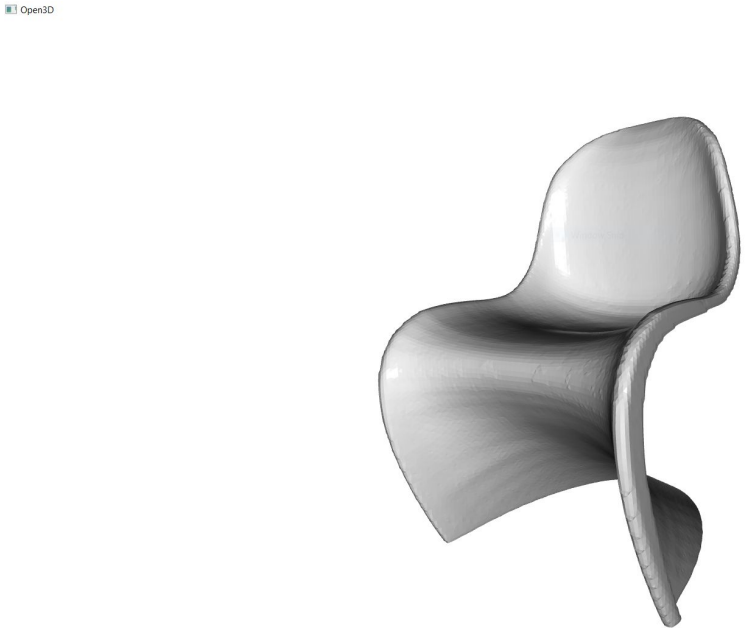

PartLabel.LEG
PartLabel.SEAT
PartLabel.BACK
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.6 −0.4 −0.2 0.0 0.2 0.4 0.6
PartLabel.LEG
PartLabel.SEAT
PartLabel.BACK
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.6 −0.4 −0.2 0.0 0.2 0.4 0.6

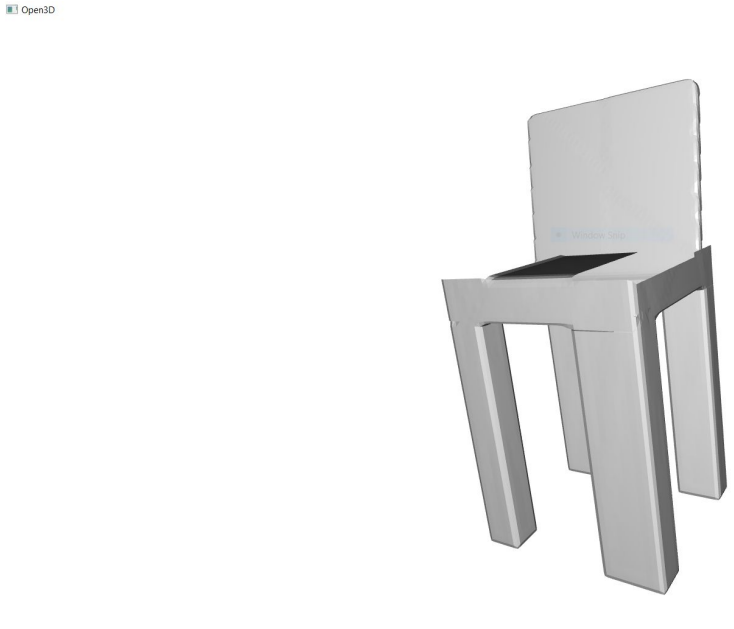
Replacement without deformation



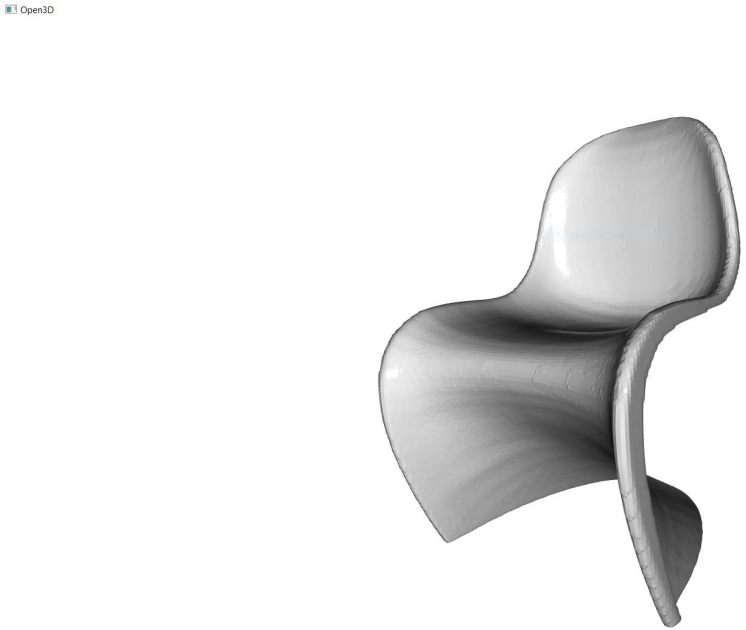


Replacement without deformation





Replacement without deformation



Part Movement

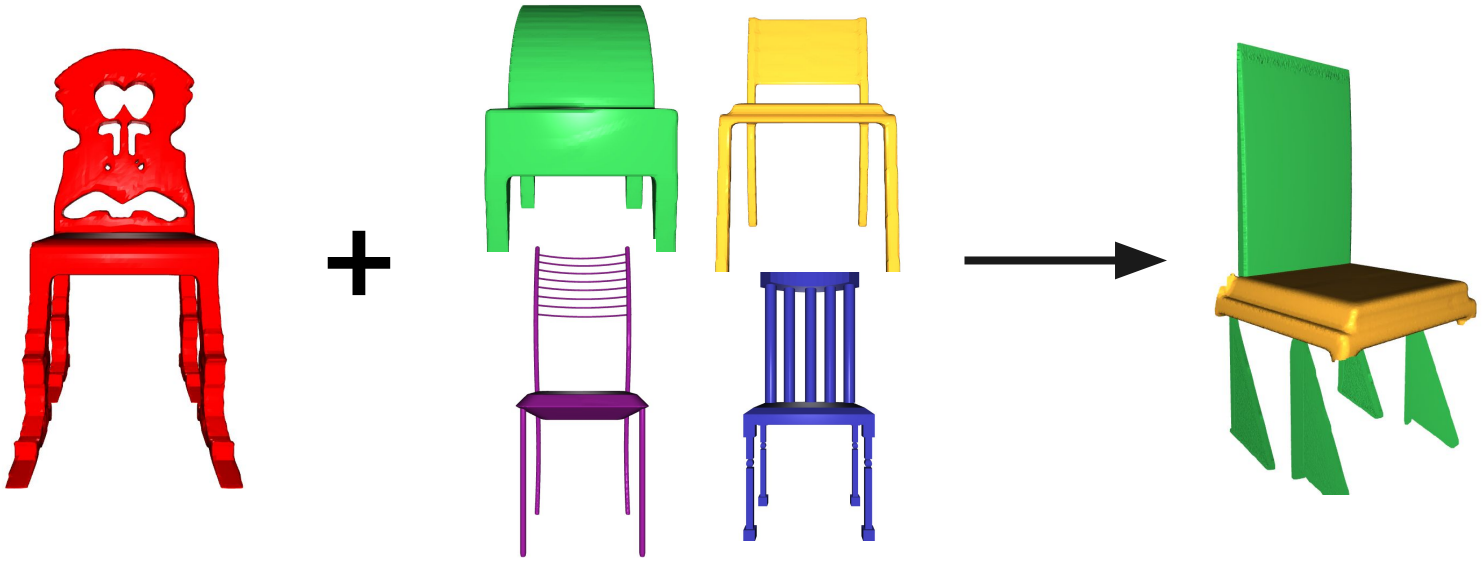
After warping, replacing and resizing, some parts were still not joined together to form a coherent chair. This was common in the case of LEG, BACK and ARMREST with respect to the SEAT part, where they would be floating slightly above (BACK AND ARMREST) or below (LEG) the seat.

To solve this, we calculate the distance of the LEG, BACK and ARMREST from the SEAT for the points. We sort them and choose the nth (by default 1000 is a good value) least value and move the parts towards the SEAT by that distance.

Part Movement Results

Before and after movement is applied

Full Result

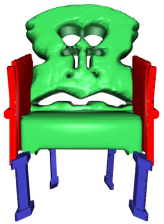


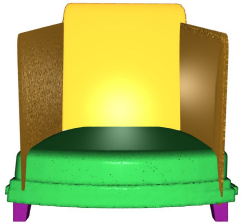
Target

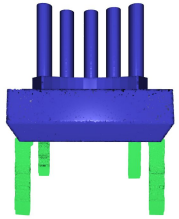
Sources

Result

Some Creative Results







Plausibility Scorer

For the plausibility scorer, we trained 3 different neural networks:

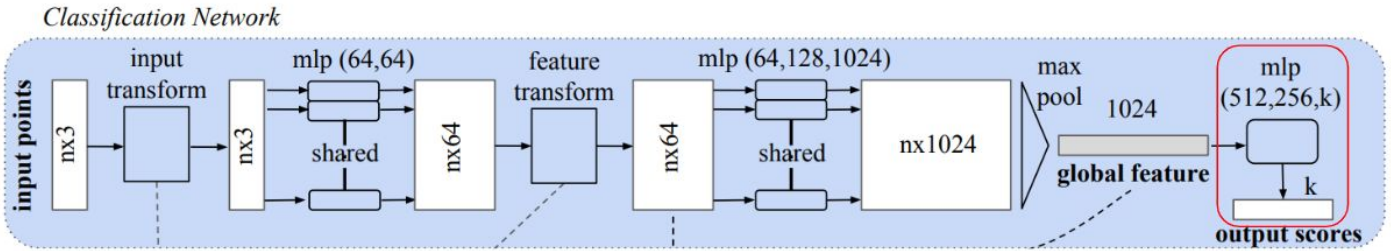
- LeNet
- ResNet
- PointNet

PointNet

As the output generated by the mixer is a point cloud, we considered PointNet as an one of the models for scoring plausibility. The input to the model is the a set of 3D point coordinates (x, y, z) sampled from the point cloud and the output is a score between 0-1.

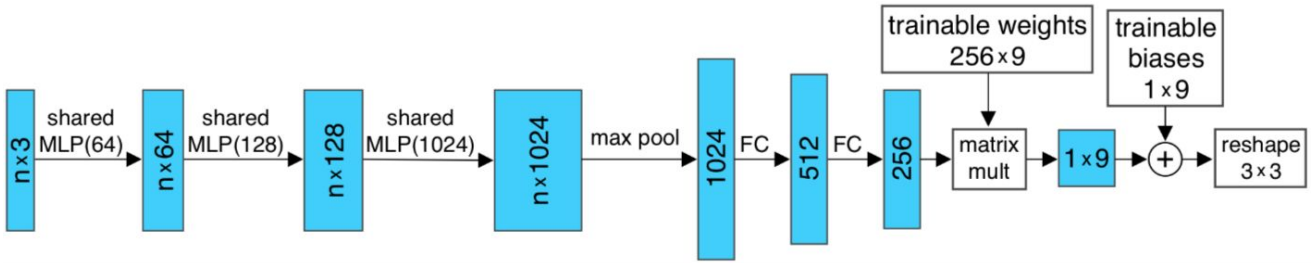
PointNet

PointNet is generally used as a classifier network, outputting ‘k’ class probabilities. We assign one of 2 labels/classes to each of our input point clouds: positive or negative.



Transformation Network (TNet)

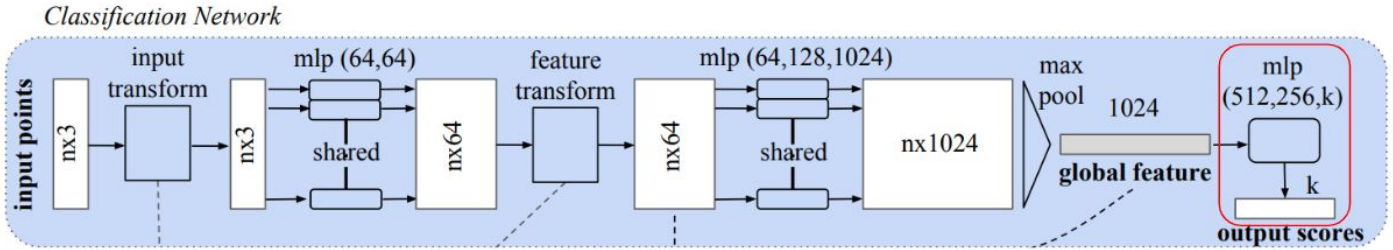
Given a point cloud, this network learns a affine transformation matrix that can be applied to the point cloud to normalize it w.r.t rotations and translations.



PointNet

Our network uses only one TNet, applied to input points.

Max-pooling is applied to obtain global features that are invariant to the order of the input points. This ensures that the classification is independent of the order of the input points.



Dataset for PointNet

We parse the model files (.obj) in the PartNet-Symh dataset to obtain the mesh vertices for various parts of the chair.

For positive examples, mesh vertices from all parts of the chair are added to the point cloud. For negative examples, mesh vertices from certain parts are skipped at random.

A fixed number of points (2500 in our case) are sampled at random from all point clouds and input to the network.

Views for Original LeNet Score Evaluation

3 views rendered with Python for the generated results (Front, Right and Top)

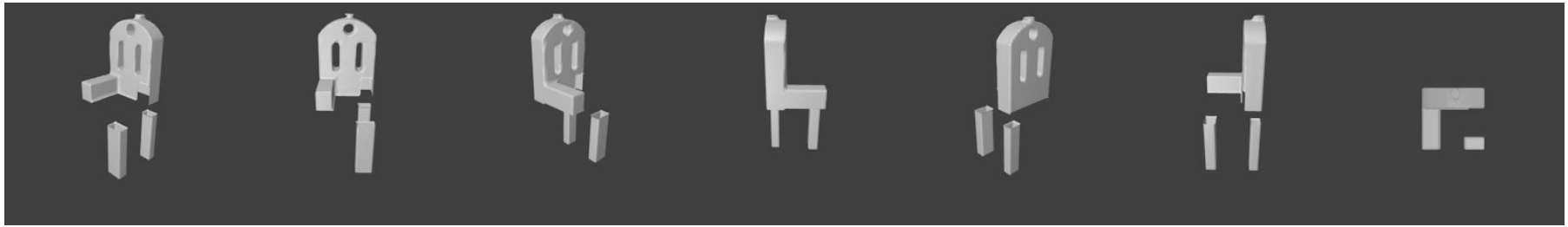
Dataset for LeNet and ResNet

7 views rendered with Blender and Python (45 side views, front and iso top)

Input

Positive class: 6201 original models renders

Negative class: 6201 models with random parts removed



Plausibility scorer using LeNet

Test Set A, B and C are all good chairs, compared to the Bad Chair dataset. When the plausibility scorer using LeNet given to us was tested using our data, a score between 0.60 to 0.75 was obtained for Set A, 0.49 to 0.78 for four of the chairs in set B and one of them getting a score of 0.27 which shows inconsistency.

Similarly, set C chairs obtained a score between 0.69 to 0.92. But the bad chair dataset obtained a score of 0.85 and 0.82 for two of the chairs.

Due to this continued inconsistency, we generated our own dataset of increased length, and four additional views. Then, we trained the LeNet model on this dataset for 20 epochs.

Probability scorer using LeNet on our dataset for 7 views

We extended the original LeNet code to support 7 views with our custom dataset. It was run for 20 epochs with a subset of 1500 models (around 10 000 views). We split the data 80/20 for training and testing. Due to time limitation the model was not fully trained and produced an average test accuracy of 70%.

Modified Plausibility Scorer using ResNet

Due to the inconsistency in the results obtained by LeNet, we opted to use ResNet for our model.

Initially, we used a pre-trained ResNet34 model with an added convolutional layer, and a fully connected layer with ReLU activation. The results were unsatisfactory with the accuracy for some of the views accounting to 0.50 only. The probability scores for the bad chairs were less than 0.40, but we obtained a score between 0.4 to 0.5 for good chairs. Since the score for good chairs must be more than 0.5, we switched to ResNet18 with additional layers added to it.

Architecture of our ResNet model

A pre-trained ResNet 18 model was taken, and a convolutional layer with kernel size 7 was added.

This sequential layer was followed by a fully connected layer with activation function ReLU. This takes the results of the convolutions above and classifies the image into a label. This is followed by a dropout layer to avoid overfitting. Finally, another fully connected layer was added to the model.

Network Structure and Dataloader

The entire scorer was re-coded to use PyTorch. While training, we iterate through every view to train a separate model for each one. The good and bad chairs are loaded from our dataset, appended into lists for each view, loaded into the train_dataloader. A tuple consisting of the image and label pair is each enumerated after pre-processing and trained for 25 epochs. The cross entropy loss function is used to measure training loss.

In the testing phase, the model for each view is loaded one by one, and probabilities are obtained for all the seven views for each chair. The minimum of the probabilities is the score for the chair, with more than 0.5 being a good chair and less than 0.5 being a bad chair.

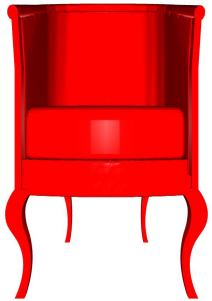
Results

For the good chairs, a score between 0.39 to 0.50 was obtained for Set A, between 0.31 to 0.60 for Set B and between 0.39 to 0.49 for Set C.

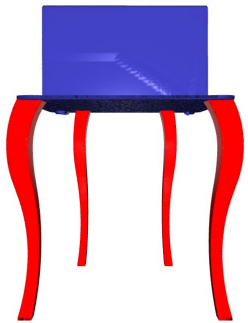
For the bad chairs, a score of about 0.30 was obtained.

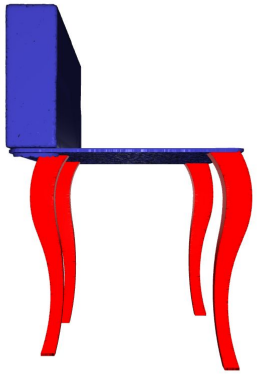
Even, though the difference in the scores between the good and bad chairs is considerable, a value above 0.5 was desired for the good chairs. Hence, we shifted our approach to PointNet.

Set A



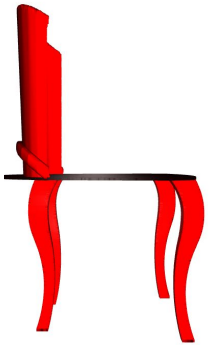
Results (Set A)





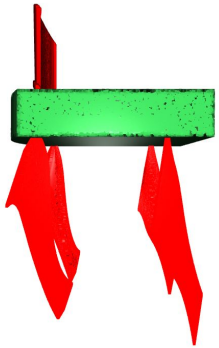
Plausibility Scores	
LeNet (given models)	0.616529
LeNet (trained on our dataset)	0.7312996
ResNet	0.378438
PointNet	0.737148

Results (Set A)



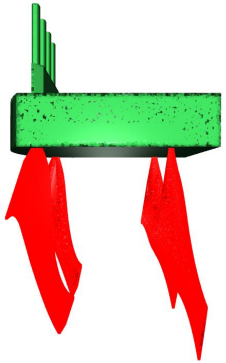
Plausibility Scores	
LeNet (given models)	0.713682
LeNet (trained on our dataset)	0.759339
ResNet	0.490273
PointNet	0.743933

Results (Set A)



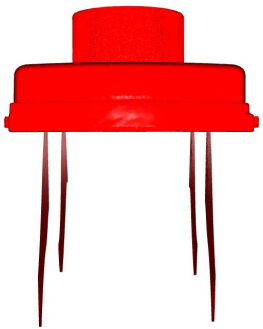
Plausibility Scores	
LeNet (given models)	0.766048
LeNet (trained on our dataset)	0.796677
ResNet	0.42773
PointNet	0.707632

Results (Set A)



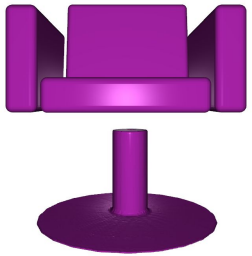
Plausibility Scores	
LeNet (given models)	0.726444
LeNet (trained on our dataset)	0.7806684
ResNet	0.419397
PointNet	0.687341

Results (Set A)



Plausibility Scores	
LeNet (given models)	0.698750
LeNet (trained on our dataset)	0.7590514
ResNet	0.432083
PointNet	0.750989

Set B

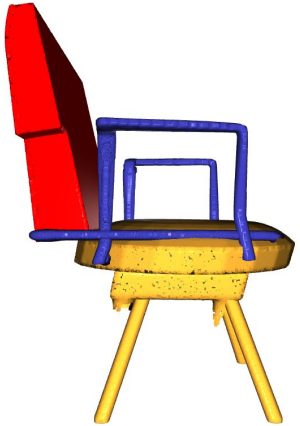


Set B

Results (Set B)

Plausibility Scores	
LeNet (given models)	0.273834
LeNet (trained on our dataset)	0.746293
ResNet	0.324794
PointNet	0.915892

Results (Set B)

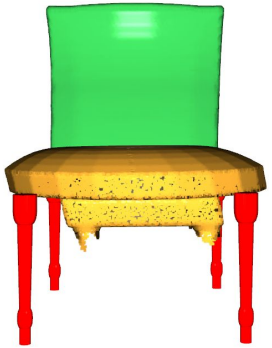


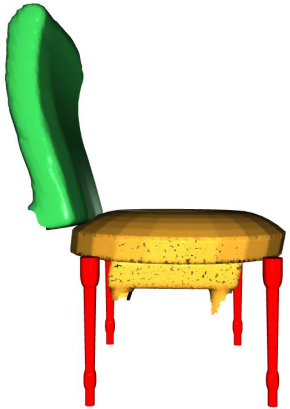
Plausibility Scores	
LeNet (given models)	0.521426
LeNet (trained on our dataset)	0.755563
ResNet	0.315487
PointNet	0.864023

Results (Set B)

Plausibility Scores	
LeNet (given models)	0.493524
LeNet (trained on our dataset)	0.613947
ResNet	0.564144
PointNet	0.758692

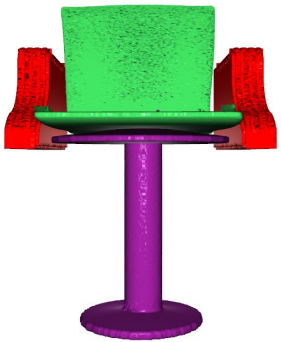
Results (Set B)





Plausibility Scores	
LeNet (given models)	0.525308
LeNet (trained on our dataset)	0.731478
ResNet	0.358845
PointNet	0.714601

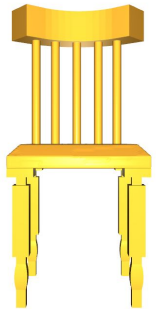
Results (Set B)

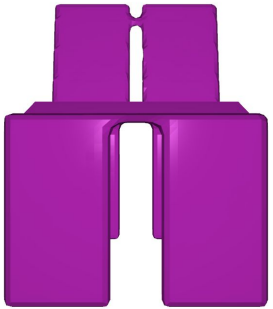


Plausibility Scores	
LeNet (given models)	0.789838
LeNet (trained on our dataset)	0.807321
ResNet	0.442619
PointNet	0.610238

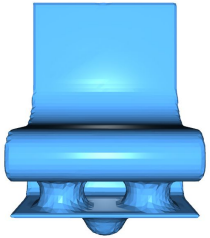
Set C

Set C

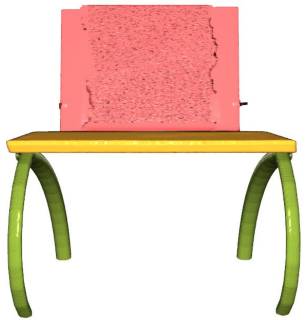


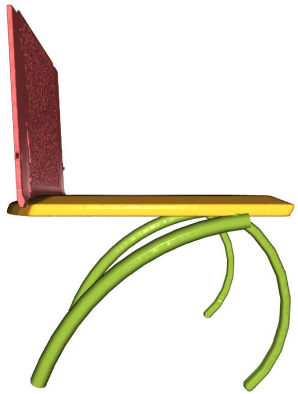


Set C



Results (Set C)



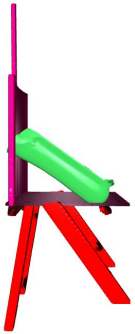


Plausibility Scores	
LeNet (given models)	0.709786
LeNet (trained on our dataset)	0.766016
ResNet	0.403124
PointNet	0.836210

Results (Set C)

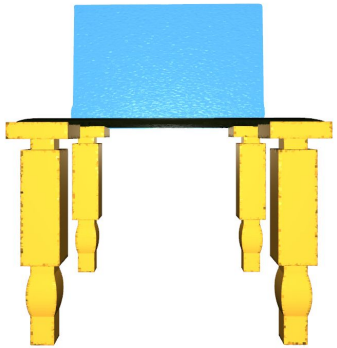
Plausibility Scores	
LeNet (given models)	0.923031
LeNet (trained on our dataset)	0.818013
ResNet	0.433721
PointNet	0.810363

Results (Set C)



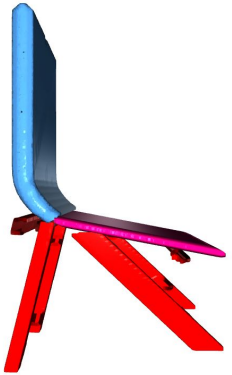
Plausibility Scores	
LeNet (given models)	0.850199
LeNet (trained on our dataset)	0.783778
ResNet	0.491123
PointNet	0.779533

Results (Set C)



Plausibility Scores	
LeNet (given models)	0.694466
LeNet (trained on our dataset)	0.760559
ResNet	0.386702
PointNet	0.729349

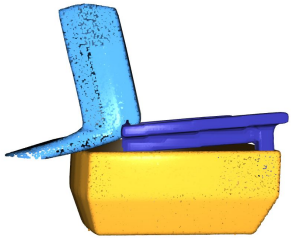
Results (Set C)



Plausibility Scores	
LeNet (given models)	0.833766
LeNet (trained on our dataset)	0.742372
ResNet	0.384734
PointNet	0.572233

Comparison Table : “Bad” Chairs

Plausibility Scores	
LeNet (given models)	0.856642
LeNet (trained on our dataset)	0.7832434
ResNet	0.375208
PointNet	0.335514



Plausibility Scores	
LeNet (given models)	0.822415
LeNet (trained on our dataset)	0.680848
ResNet	0.316534
PointNet	0.198797

Plausibility Scores	
LeNet (given models)	0.682733
LeNet (trained on our dataset)	0.737367
ResNet	0.43816
PointNet	0.055074

Conclusion

- LeNet is unable to clearly distinguish between a plausible chair and chairs with missing or unusually deformed parts (“bad” chairs). Both types of chairs are given high plausibility scores.
- ResNet scores “bad” chairs low, but does not give a much higher score to plausible chairs.
- PointNet can make a better distinction in plausibility of the generated chairs. Plausible chairs are given a high score, while “bad” chairs are given low scores.

References

1. Fenggen Yu et al.: A Recursive Part Decomposition Network for Fine-grained and Hierarchical Shape Segmentation, 2019
2. Kaichun Mo et al. : A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding, 2019
3. Charles R. Qi et al.: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation, 2016
4. Andriy Myronenko et al.: Point-Set Registration: Coherent Point Drift, 2009