

From g16

3. For structure approach, what are the reasons for choosing panoramic view features instead of the 1-hop neighborhood ring features? They both seem to have rotational invariance property.

The 1-hop neighborhood ring features method requires a fine mesh that each triangle have a similar size, and a relatively smooth surface. However, the ModelNet10 include objects like bed and dressers which contain sharpe edges.

From g20

-The preprocessing part speed is a little long.

Yes, it is. I still didn't find a way to make it faster. Although the preprocessing is already speeded up using hardware acceleration (gpuArray and parallel computing), it still takes about 10 minutes for ModelNet10 set.

From g73

● A little recommendation is that the length of the result in your presentation can be introduced longer and more complete as your previous slides. For example, why will the accuracy in a random rotated test decrease dramatically?

Because the first method does not preserve rotational invariance. Once rotated, even a small angle will make all layers in the 3d matrix completely different. The nature of voxelization also limits us from extract or generate more train data because we have to do full rotations to all vertices of each model and regenerate voxels, which means we can expend our train set with low cost. This problem does not exist in second method, because to generate expended rotation set for cylindrical views, we can simply shift the 2d image right or left, which can be done super fast, then by using the expended set as train set, we can make the model robust though the input is rotated.

Group 37: 3D Mesh Classification

Zeyu Duan

UCSD ECE 228

zeduan@ucsd.edu

Abstract—This electronic document is a “live” template and already defines the components of your paper [title, text, heads, etc.] in its style sheet. ***CRITICAL: Do Not Use Symbols, Special Characters, Footnotes, or Math in Paper Title or Abstract. (Abstract)**

Keywords—component, formatting, style, styling, insert (key words)

I. INTRODUCTION (HEADING 1)

This project focuses on classification of 3D object data, specifically the mesh data. Because of the non-Euclidean nature of 3d mesh data, it is hard to apply the CNN or other models directly. A common approach is to develop algorithms that suitable for non-Euclidean mesh data, however, many of these algorithms require the input mesh to be a fine mesh, such that the algorithm can build local patches crossing several triangles that have similar size. Another approach is to convert the mesh data to Euclidean data and then process it with classical CNNs, such as multi-view approach.

In this project, two types of conversions and corresponding networks are applied. One is a simple voxelization method that converts meshes to [33x33x33] grids and train them with CNNs. The second one is a more complex approach that convert the data to a more robust and rotation invariance representation.

II. RELEATED WORK

A. Multi-view Convolution

This type of methods converts the model into a series of pictures, and then train them as training an image classification model. This article [1] introduces an implementation of multi-view approach. The team take a series of picture around each object, those images were sent to a multi-CNN network which is followed by a view pooling. The final result shows great accuracy, but this method is computational expensive, it requires to render dozens of pictures for a single object, this process will take a long period when we have fine models which might contain too many faces.

B. Voxolization

This approach converts the meshes or point clouds into 3d grids, and the 3d grids contains values 0 or 1 to imply if this cell is occupied by the model. To yield better results, the 0 and 1 might be replaced by float values that indicate how much

percent of this grid is occupied. This method is simple precise and intuitive, also very compatible with CNNs we have right now, but a big problem of this approach is robustness. This type of methods require model to be well aligned, a small angle along any axis might lead to a huge drop on the accuracy.

III. DATA SET

The dataset used in this project is the ModelNet10 dataset, which is presented by Princeton ModelNet.

This dataset includes 10 categories objects, int total 3991 train models and 908 test models

Besides the 908 test models, we made a copy of this test set, each model in this new set is oriented at a random angle between 0 to 90 degrees along the gravitational axis. This new set is then used to test the robustness of networks.

A. For Voxel Method

In this approach, each model is voxelized into [33x33x33] grid and arranged into a big array. The grid is generated by subdividing triangle in the mesh until no edges is longer than a certain value. Then matching the position of vertex to the grids, if there exist more than one vertex in a grid cell, we set corresponding value in the array to 1, otherwise 0. Figure 1 shows two models and their voxels.

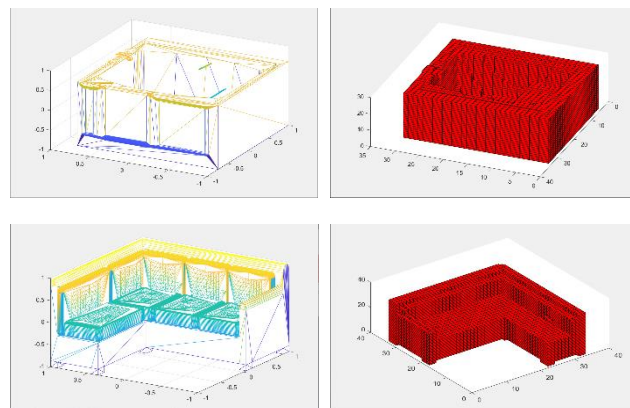


Figure 1

B. Cylindrical View

This approach builds a virtual cylinder surface around the object with radius equals to the max distance from vertices to z axis.

We then slice the surface to 40 layers from top to bottom, and each layer is subdivided into 112 degrees. Now we have [40x112] regions. Each region will contain a value between [0,1] that describes how close this region is to the closest point on the orthogonal ray from this region to the z axis that intersection with the mesh (figure 2).

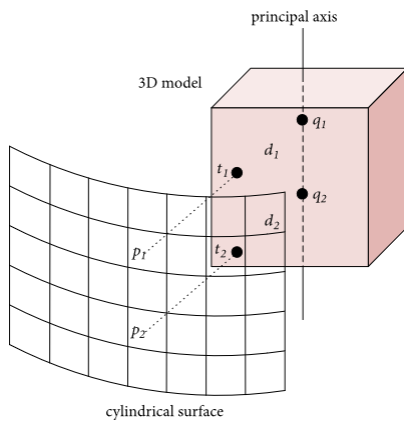


Figure 2 [2]

A model and its 2d representation might look like Figure 3.

Now we can process the mappings as normal 2d grayscale images using classical CNNs.

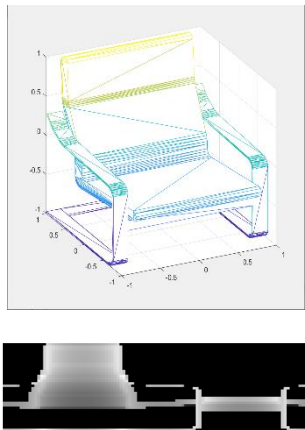


Figure 3

IV. METHODS

A. Voxel – from VoxNet [1]

The input to the network is a [33x33x33] 1 channel 3d image, followed by two convolutional layers. The first one has 32 kernels each with [5x5x5] kernel size, the second one has 32

kernels each with [3x3x3] kernel size. More specific structures:

3-D Image Input	33×33×33×1
Convolution	33×33×33×32
Dropout	33×33×33×32
Convolution	33×33×33×32
3-D Max Pooling	16×16×16×32
Dropout	16×16×16×32
Fully Connected	1×1×1×1280
Dropout	1×1×1×1280
Fully Connected	1×1×1×10
Softmax	1×1×1×10
Classification Output	-

B. Cylindrical View [3]

Here we need a more complex model structure that can handle the robustness and rotational invariance. Therefore, this model contains 4 convolutional layers. Each convolution layer is followed by a max pooling layer of size [2x2] and a dropout layer of rate = 0.2. Before the softmax function, we have 2 fully connect layers with 512 and 1024 hidden units.

However, the structure itself cannot provide a robust result. To make the model more robust, we need to expend the input data. This idea is posted in article² [3]. Normally, we need to recalculate the 2d mappings of each object in different angles to make the model robust, this process will take too long if we rotate the whole surface (or model) and regenerate mappings. However, this can be done much easier and faster if we just shift the original mappings we have, because rotating the model and recompute will produce the same result as just shifting the pixels

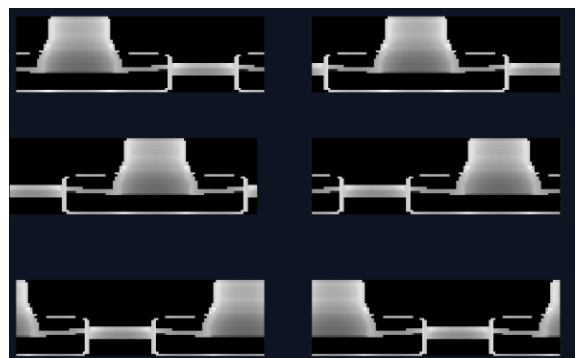


Figure 4

on the 2d mappings right of left. In figure 4 we illustrate how the rotated mappings of the same chair look like.

Layers	Size
Input	[40x112x1] grayscale image
Conv1	64 [1x1] kernels
Conv2	80 [2x2] kernels with Stride [2x2]
Conv3	160 [2x2] kernels with Stride [2x2]
Conv4	320 [2x2] kernels with Stride [2x2]
FC	512 hidden units
FC	1024 hidden unit
Softmax	10 categories
Classification	1 output

The method above makes it possible to generate large amount of training data in different angles, so we choose to expend the original set to **28x** larger, which means rotate each model 12.8 degrees to the left, or 4 pixels to the right for each 2d map. For the training set that contains 3991 models, this process can be done in less than 5 seconds.

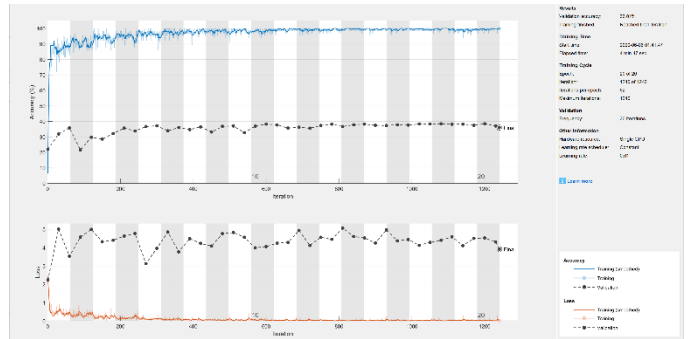
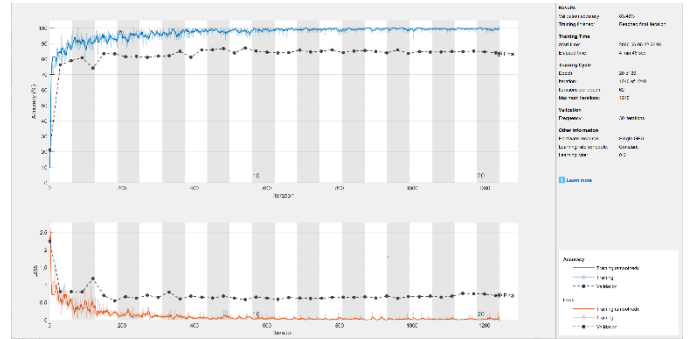
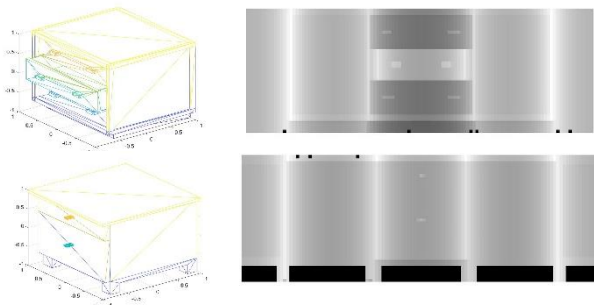
V. RESULT

All result is generated with learning rate at 0.01 an Mini Batch Size = 64.

The result of first method is pretty good when the models are perfectly aligned, this simple method can achieve around 85% accuracy. However, the accuracy on randomly rotated test set is very low, only about 35%, this gap indicates that the voxel model is not robust. Figure 5 to 7 show the accuracy of method 1 on original set, accuracy of method 1 on rotated set and the confusion matrix of original test set respectively.

Method 2 shows great robustness on rotated test set. It achieves **80.1%** total accuracy on original set, and **80.4%** on the rotated set. Although the accuracy on well aligned set drops slightly, the model is now much more stable on most inputs. Figure 8 and 9 show the confusion matrix of original test set and rotated test set.

Although the first method does not show robustness, some common features are noticed in both results. Both of them cannot clearly identify the difference between “night stand” type objects and “dresser”. By comparing the models and 2d representations of these two objects, we notice that they are very similar no matter on structures or the cylindrical views. The classification of such object might be a topic of future work.



Output Class \ Target Class	bathtub	bed	chair	desk	dresser	monitor	night_stand	sofa	table	toilet	Accuracy
bathtub	39 4.3%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	0 0.0%	0 0.0%	0 0.0%	2 0.2%	90.7%
bed	5 0.6%	92 10.1%	0 0.0%	2 0.2%	2 0.2%	2 0.2%	2 0.2%	1 0.1%	0 0.0%	2 0.2%	85.2%
chair	0 0.0%	5 0.6%	100 11.0%	1 0.1%	0 0.0%	2 0.2%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	92.6%
desk	0 0.0%	0 0.0%	0 0.0%	62 6.8%	2 0.2%	1 0.1%	2 0.2%	0 0.0%	0 0.0%	6 0.7%	84.9%
dresser	4 0.4%	2 0.2%	0 0.0%	4 0.4%	72 7.9%	2 0.2%	31 3.4%	2 0.2%	0 0.0%	0 0.0%	61.5%
monitor	1 0.1%	0 0.0%	0 0.0%	0 0.0%	1 0.1%	90 9.9%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	97.8%
night_stand	0 0.0%	0 0.0%	0 0.0%	1 0.1%	9 1.0%	0 0.0%	35 3.9%	0 0.0%	0 0.0%	0 0.0%	77.8%
sofa	0 0.0%	0 0.0%	0 0.0%	3 0.3%	0 0.0%	2 0.2%	0 0.0%	97 10.7%	0 0.0%	0 0.0%	95.1%
table	0 0.0%	0 0.0%	0 0.0%	13 1.4%	0 0.0%	0 0.0%	16 1.8%	0 0.0%	94 10.4%	0 0.0%	76.4%
toilet	1 0.1%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	0 0.0%	96 10.6%	99.0%
Total	78.0%	92.0%	100%	72.1%	83.7%	90.0%	40.7%	97.0%	94.0%	96.0%	85.6%
Rotated	22.0%	8.0%	0.0%	27.9%	16.3%	10.0%	59.3%	3.0%	6.0%	4.0%	14.4%

Target Class
Figure 5 to 7

VI. CONCLUSION

In general, the second method performs much better, because we cannot promise the model will always be aligned perfectly, and method 2 shows great robustness and rotational invariance. However, the drawback of second method is the lack of information about internal structure. In other words, we actually waste a lot of information intentionally, and a advantage of having a 3d mesh is that meshes can deliver more information than 2d images. Therefore, the future work might focus on non-Euclidean methods, so that we don't have to give up information intentionally.

VII. REFERENCES

[1]. D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In IROS, 2015.

[2].

<https://www.hindawi.com/journals/mpe/2018/6467957/fig2/>

[3]. Qiang Zheng, Jian Sun, Le Zhang, Wei Chen, Huanhuan FanAn. Improved 3D Shape Recognition Method Based on Panoramic View

Toolbox references:

[1]. Voxelization:

<https://www.mathworks.com/matlabcentral/fileexchange/2408-6-polygon2voxel>

[2]. Hardware Accelerated Ray-triangle intersection

<https://www.mathworks.com/matlabcentral/fileexchange/4967-0-hardware-accelerated-ray-triangle-intersection>

[3]. Matlab parallel computing toolbox

[4]. Matlab Deep learning toolbox

[5]. Matlab Image processing toolbox

Output Class	bathtub	bed	chair	desk	dresser	monitor	night_stand	sofa	table	toilet	
bathtub	1006 4.0%	121 0.5%	0 0.0%	41 0.2%	141 0.6%	59 0.2%	96 0.4%	0 0.0%	0 0.0%	49 0.2%	66.5%
bed	88 0.3%	2430 9.6%	188 0.7%	29 0.1%	51 0.2%	0 0.0%	79 0.3%	221 0.9%	3 0.0%	105 0.4%	76.1%
chair	10 0.0%	71 0.3%	2463 9.7%	184 0.7%	28 0.1%	28 0.1%	36 0.1%	154 0.6%	0 0.0%	0 0.0%	82.8%
desk	0 0.0%	0 0.0%	0 0.0%	1684 6.6%	17 0.1%	0 0.0%	132 0.5%	3 0.0%	625 2.5%	0 0.0%	68.4%
dresser	193 0.8%	7 0.0%	0 0.0%	4 0.0%	1969 7.7%	0 0.0%	794 3.1%	15 0.1%	1 0.0%	6 0.0%	65.9%
monitor	4 0.0%	57 0.2%	29 0.1%	108 0.4%	22 0.1%	2685 10.6%	5 0.0%	17 0.1%	26 0.1%	16 0.1%	90.4%
night_stand	24 0.1%	0 0.0%	13 0.1%	35 0.1%	151 0.6%	0 0.0%	1072 4.2%	0 0.0%	40 0.2%	0 0.0%	80.3%
sofa	2 0.0%	40 0.2%	79 0.3%	85 0.3%	0 0.0%	28 0.1%	7 0.0%	2327 9.2%	0 0.0%	0 0.0%	90.6%
table	0 0.0%	0 0.0%	1 0.0%	187 0.7%	0 0.0%	0 0.0%	174 0.7%	0 0.0%	2105 8.3%	0 0.0%	85.3%
toilet	73 0.3%	74 0.3%	27 0.1%	51 0.2%	29 0.1%	0 0.0%	13 0.1%	63 0.2%	0 0.0%	2624 10.3%	88.8%
	71.9% 28.1%	86.8% 13.2%	88.0% 12.0%	69.9% 30.1%	81.8% 18.2%	95.9% 4.1%	44.5% 55.5%	83.1% 16.9%	75.2% 24.8%	93.7% 6.3%	80.1% 19.9%

Output Class	bathtub	bed	chair	desk	dresser	monitor	night_stand	sofa	table	toilet	
bathtub	1158 4.6%	80 0.3%	0 0.0%	85 0.3%	257 1.0%	56 0.2%	135 0.5%	6 0.0%	0 0.0%	70 0.3%	62.7%
bed	84 0.3%	2443 9.6%	113 0.4%	28 0.1%	67 0.3%	0 0.0%	3 0.0%	177 0.7%	0 0.0%	61 0.2%	82.1%
chair	6 0.0%	82 0.3%	2615 10.3%	173 0.7%	28 0.1%	30 0.1%	86 0.3%	176 0.7%	0 0.0%	32 0.1%	81.0%
desk	0 0.0%	4 0.0%	0 0.0%	1637 6.4%	26 0.1%	0 0.0%	143 0.6%	0 0.0%	567 2.2%	0 0.0%	68.9%
dresser	68 0.3%	6 0.0%	0 0.0%	25 0.1%	1739 6.8%	0 0.0%	687 2.7%	0 0.0%	15 0.1%	0 0.0%	68.5%
monitor	21 0.1%	62 0.2%	1 0.0%	33 0.1%	28 0.1%	2686 10.6%	0 0.0%	6 0.0%	28 0.1%	0 0.0%	93.8%
night_stand	0 0.0%	25 0.1%	27 0.1%	45 0.2%	245 1.0%	0 0.0%	955 3.8%	0 0.0%	6 0.0%	0 0.0%	73.3%
sofa	20 0.1%	58 0.2%	36 0.1%	102 0.4%	18 0.1%	28 0.1%	21 0.1%	2384 9.4%	0 0.0%	0 0.0%	89.4%
table	0 0.0%	0 0.0%	0 0.0%	256 1.0%	0 0.0%	0 0.0%	362 1.4%	19 0.1%	2184 8.6%	0 0.0%	77.4%
toilet	43 0.2%	40 0.2%	8 0.0%	24 0.1%	0 0.0%	0 0.0%	16 0.1%	32 0.1%	0 0.0%	2637 10.4%	94.2%
	82.7% 17.3%	87.3% 12.7%	93.4% 6.6%	68.0% 32.0%	72.2% 27.8%	95.9% 4.1%	39.7% 60.3%	85.1% 14.9%	78.0% 22.0%	94.2% 5.8%	80.4% 19.6%

Figure 8 to 9