PointGrid: A Deep Network for 3D Shape Understanding

Truc Le and Ye Duan
University of Missouri – Columbia
tdlxqb@mail.missouri.edu, duanye@missouri.edu

Abstract

Volumetric grid is widely used for 3D deep learning due to its regularity. However the use of relatively lower order local approximation functions such as piece-wise constant function (occupancy grid) or piece-wise linear function (distance field) to approximate 3D shape means that it needs a very high-resolution grid to represent finer geometry details, which could be memory and computationally inefficient. In this work, we propose the PointGrid, a 3D convolutional network that incorporates a constant number of points within each grid cell thus allowing the network to learn higher order local approximation functions that could better represent the local geometry shape details. With experiments on popular shape recognition benchmarks, PointGrid demonstrates state-of-the-art performance over existing deep learning methods on both classification and segmentation.

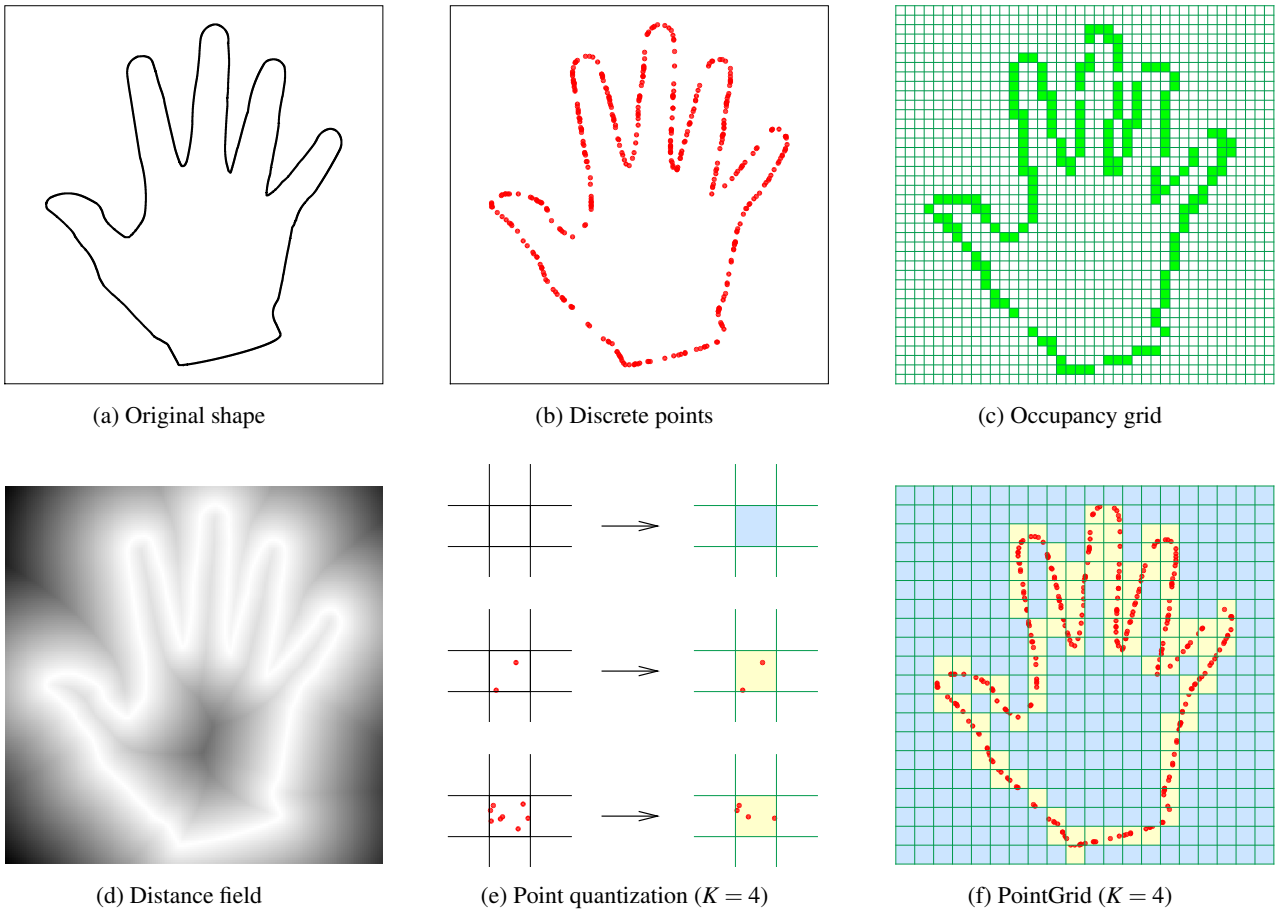
1. Introduction

Deep learning has become a universal tool for many visual recognition tasks ranging from classification to segmentation, especially ConvNets for 2D images [28, 48, 50, 16, 10, 34, 37, 45, 18] thanks to its weight sharing and other kernel optimizations of 2D convolutions. It is therefore natural that a lot of researchers currently aim at the adaptation of deep ConvNets to 3D models. Such adaptation is, however, non-trivial due to the nature of 3D data representations. Currently the 3D geometry shape representation consists of point, mesh and volumetric grid. Mesh is extremely irregular and hence it is very hard to design a framework to directly learn from it. Point is flexible but it is unorganized. Volumetric grid is regular, which enables many researchers to utilize either occupancy grid or distance field as a mean of data representation and learn 3D convolutional networks from it.

Belonging to the volumetric grid, VoxNet and its variants [57, 35, 54, 6, 39, 32] is the most straightforward approach which transforms a 3D model into an occupancy grid. However, naive implementation of VoxNet does not scale well for dense 3D data because computational and memory requirements grow cubicly with the 3D grid resolution. A typical VoxNet takes as input a grid of size $64 \times 64 \times 64$ which is incapable of exploiting the rich and detailed geometry of the original 3D data. To resolve these issues, Kd-Net [27], O-CNN [55] and Oct-Net [42] in many respects mimic ConvNets but use the kd-tree or oct-tree structure to form the computational graph and apply 3D convolutions level by level, to share the learnable parameters, and to compute a sequence of hierarchical representations in a feedforward bottom-up fashion. These approaches exploit the sparsity of 3D data and can adaptively allocate computational and memory resources with respect to the data density. However, due to the use of more complicated data structures, it is generally not a simple task to implement these networks efficiently.

Recently Qi et al. [38] proposed PointNet that can consume unorganized point sets in 3D. In this network, all 3D points share the same set of multi-layer perceptrons which independently transform individual points. However, a single max-pooling layer is the only global operation in PointNet, which limits its ability to examine contextual neighborhood structure of the points.

In this work, we propose the *PointGrid*, a 3D convolutional network that is an integration of point and grid, a hybrid model that can better represent the local geometry shape details (Fig. 1). The proposed method scales better than volumetric grid and avoid information loss at the same time. PointGrid has an embedding volumetric grid that has the regular structure which allows 3D convolutions to extract global information hierarchically. In each grid cell, we sample a constant number of points (e.g. K) to overcome the grid size limitation. We expect the sampling points within the grid cell can better represent the local geometry shape details while the grid scales well with respect to data size as it only scales linearly in K, not cubicly as in pure volumetric grid. Later, we also show that PointGrid does not require a high resolution grid to perform well and a grid of $16 \times 16 \times 16$ is experimentally sufficient, which is substantially smaller than a typical $64 \times 64 \times 64$ grid of VoxNet. As a result, PointGrid (see Fig. 2) is simpler and faster in



(a) Original shape (b) Discrete points (c) Occupancy grid

(d) Distance field (e) Point quantization ($K = 4$) (f) PointGrid ($K = 4$)

Figure 1: A 2D illustration of the proposed PointGrid, which is a hybrid 3D shape representation between discrete points (b) and volumetric grid (c) and (d). Points within each grid cell will be quantitized (e), so that both occupied (yellow) and empty (blue) cells have exactly K points (f).

both training and testing. By experiments, PointGrid compared favorably with state-of-art methods including PointNet [38], PointNet++ [40], Kd-Net [27], O-CNN [55] and Oct-Net [42], with a smaller memory footprint.

2. Related Work

This section briefly goes over some of the existing approaches for 3D classification and segmentation, which can generally be categorized into two classes: learning on hand-crafted features or deep learning.

Hand-crafted features: The traditional approaches typically first extract local features such as planarity of various forms, higher degree geometric proxies (cylinders, cones, spheres, etc.), dihedral angles between triangles [58], curvatures (Gaussian curvature or mean curvature) [30], geodesic distances on a mesh [17, 64], slippage [61], symmetry, convexity, medial axis [33], shape diameter [44] and motion characteristics [43], shape contexts [4], spin images [21], etc. After that, people either directly apply machine learning approaches on these features (e.g. k-NN, random forest, SVM [14], JointBoost classifier [23], correspondence analysis [56, 62]) or employ some local and greedy methods such as region growing [52, 20], hierarchical clustering [3, 11, 24], spectral clustering [46], k-means [60], normalized cut [13], random walk [29] and heat walk [5]. Shamir et al. [43], Agathos et al. [2] and Theologou et al. [51] gave a comprehensive overview of methodologies in 3D segmentation. In general, these approaches are often based on certain prior assumptions of some particular property of the 3D object and hence may not generalize well.

Deep learning: While deep learning has been very popular in 2D images for many years, it has just been applied in 3D recently because unlike pixels in 2D images, 3D objects do not have regular structure. As a result, in the early period, people use deep learning as a tool to learn high level features from low level cues (usually hand-crafted). The unsupervised shape segmentation proposed by Shu et al. [47] starts by over-segmenting the input model, comput-

ing patch-based local features and then uses stacked auto-encoder to learn high level features followed by Graph-Cut based segmentation. Guo et al. [15] compute local features at different scales for each triangle and arrange them into a rectangular image, which is fed forward through a convolutional neural network (CNN) to predict the semantic label for each triangle. Although these two frameworks use deep learning techniques (stacked auto-encoder, CNN) to learn high level features from local low level ones, they do not exploit the full potential of deep learning.

Recently researchers have started to either transform 3D data into regular form or refined convolution operations to adapt to the 3D's irregularity. Voxelization and multi-view are the most common representatives of the former approach. VoxNet and its variants [57, 35, 54, 6, 39, 32] discretizes the 3D bounding box into 3D occupancy grid, then applies 3D convolutions in a similar way as in 2D images. Among these approaches, VRN [6] uses Voxception-ResNet which mimics the Inception [50] and ResNet [16] and achieves state-of-the-art results for 3D object classifcation. The main drawback of the volumetric approach is the information loss due to the voxelization as well as memory and computation consumptions as they increases cubicly with respect to the voxel's resolution. Kd-Net [27] and Octree-Net [55, 42] are designed to resolve them by skipping the computation on empty cells and focusing on informative ones. However, these networks are hard to be implemented efficiently.

Su et al. [49] was the first one to apply multi-view convolutional neural network (MV-CNN) for 3D recognition. The 3D shape is rendered in multiple views, each of which is passed through an identical image-based CNN. Features obtained from multiple views are combined via a view pooling (which is the max-pooling) and then passed through another CNN to predict the final object label. Xie et al. [59] used multi-view depth images via extreme learning machine to generate per-view segmentation and combine them via Graph-Cut. This method works pretty fast due to the easy training of the extreme learning machine but it does not give high accuracy. Later, Kalogerakis et al. [22] proposed a more complete multi-view framework. They first render the 3D model with different views, each of which is fed through a shared CNN before unprojected back to 3D. The label consistency is solved by a Conditional Random Field (CRF), which is part of the network and is optimized in an end-to-end manner. Le et al. [31] proposed a MV-RNN approach which treats multi-view images as a temporal sequence and uses recurrent neural network to correlate them. Although multi-view approaches generally give compelling results, it has several limitations. First, we need to carefully choose the rendering pipeline such as image resolution with respect to data sampling density, lighting, blending and keep track of the camera parameters. Second, each view only contains partial information and it is not trivial to correlate across views. Third, multi-view approaches are limited to model only the object's surface and cannot capture 3D internal structures.

Representatives for the later direction of adapting to the 3D irregularity include PointNet [38], PointNet++ [40], SpecCNN [63]. Su et al. [38] proposed PointNet as the first neural network which directly consumes 3D point clouds. PointNet is pretty fast and robust to rigid transformation and points' ordering. Its main limitation is relying on only the max-pooling to have context information. Later, PointNet++ was developed to compensate this weakness. Yi et al. [63] proposed SpecCNN uniquely designed for polygonal mesh. SpecCNN converts convolution to multiplication in spectral domain by Fourier analysis.

3. PointGrid

Volumetric grid is widely used for 3D deep learning due to its regularity. However the use of relatively lower order local approximation functions such as piece-wise constant function (occupancy grid) or piece-wise linear function (distance field) to approximate 3D shape means it needs a very high-resolution grid to represent finer geometry details, which could be memory and computationally inefficient.

In this work, we propose the PointGrid, a 3D convolutional network that incorporates a constant number of points within each grid cell thus allowing the network to learn higher order local approximation functions that could better represent the local geometry shape details (Fig. 1).

We now introduce our PointGrid, starting with the discussion of its input format, then discussing its architecture for classification, and finally discussing how to use it for semantic segmentation of 3D point clouds.

3.1. Input Layer

The new deep architecture works with 3D grid constructed for 3D point clouds. We normalize the point cloud to the unit box $[-1,1]^3$ and this is the only preprocessing step in our framework. In contrast to VoxNet which uses occupancy grid as the primary representation of the 3D structure, we stack the points' coordinates as features for each cell. As a result, a cell with K points will have features $3K$ for the corresponding x, y and z coordinates. However, each cell has different number of points and constructing such grid is infeasible for sharing 3D convolutional kernels.

To solve this issue, we use a simple yet effective sampling strategy which we call *Point Quantization*, to keep a fixed number of K points in each cell. More specifically, if there are more than K points, we randomly sample K of them. If there are less than K points, we sample with replacement K of them. We pad zeros to cells with no point.

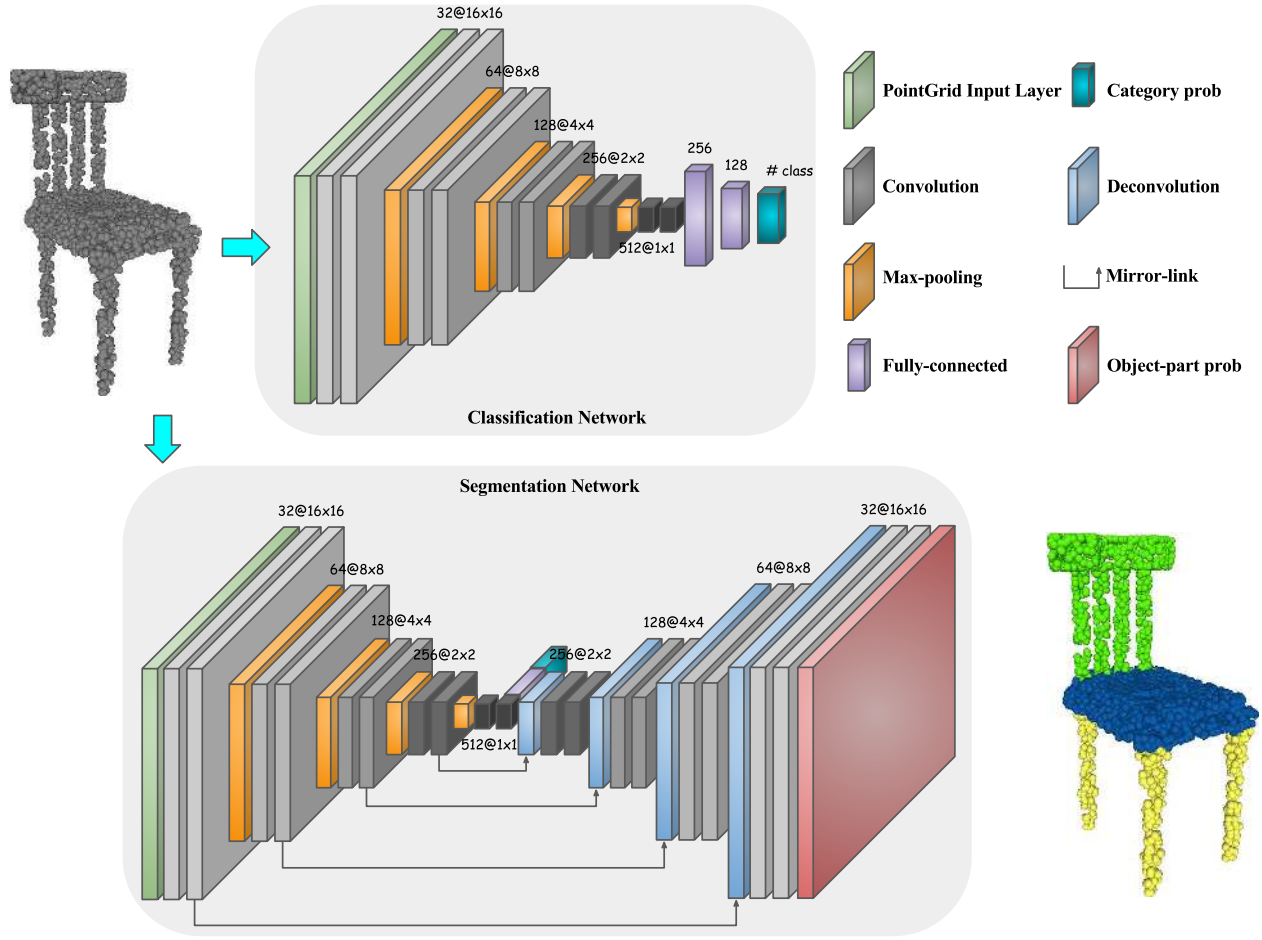


Figure 2: The architecture of PointGrid. Starting from the grid obtained from our sampling strategy, both classification and segmentation networks share the feature extraction (encoder) part. Segmentation network uses skip connections to preserve information of different hierarchical levels. All convolutions, deconvolutions and fully connected layers include batch normalization and ReLU (except object category and object-part segmentation layers). The notion 32@16x16 means there are 32 convolutional filters and the spatial dimension is 16×16. The network is visualized in 2D.

This mimics the commonly used zero-padding to compensate the boundary loss in convolutions. The value of K theoretically can be approximated by the total number of points (P) divided by the number of cells in the grid (N^3) (i.e. $K \approx \frac{P}{N^3}$). When working with point clouds of size $P = 1024$, we empirically set $K = 4$ with a grid size of $16 \times 16 \times 16$ for the best trade-off between accuracy and performance.

3.2. Classification Network

The classification network of PointGrid is illustrated in 2D in Fig. 2, which extracts features from input grid. Hence, we need the architecture to be deep and efficiently generate perceptually multi-level features. PointGrid consists of several blocks of convolutions followed by max-pooling to represent different hierarchical feature representations. Each convolution layer includes a $3 \times 3 \times 3$ kernel with stride 1 convolution, a batch normalization [19] and a rectified linear unit (ReLU) [36, 28]. The first block use 32-filter convolutions and they are doubled in each successive block. The pooling layers not only provide another form of translation invariance but also serve to progressively reduce the spatial size of the representation, to reduce the number of parameters and amount of computation in the network, and hence to also control overfitting. All of our pooling layers are max-pooling which reduce the grid size half in each spatial dimension. After several convolutional and max pooling layers, as usual, the high-level reasoning in our PointGrid is done via fully connected layers. PointGrid has two fully connected layers, each of which consists of a fully connected layer, a ReLU activation and a dropout layer (with dropout rate 0.3). Finally, another fully connected layer followed by a softmax to regress to the proba-

bility of each category. The number of nodes in this layer equals to the number of object categories in the dataset.

3.3. Segmentation Network

Our segmentation network shares the feature extraction (or encoder) from the classification network (see Fig. 2) and decodes the extracted features to build the segmentation. It is intuitive that labeling object parts depends on the object category so we include both the high-level features in the last fully connected layer and the object category probability for better global features. The decoder is almost symmetric to the encoder with convolutions replaced by deconvolutions (or sometimes referred as transposed convolutions). We optimize both networks simultaneously. The final loss is a linear combination of classification loss and segmentation loss.

The classification network progressively extracts and down-samples features, while the segmentation counterpart upsamples and combines them to construct the output. The sizes of feature maps are exactly mirrored in our network. We link early encoded features (from the classification network) to the corresponding decoded features (from the segmentation network) at the same spatial resolution, in order to obtain local sharp details preserved in early encoder layers. A mirror-link is a short notation for concatenation a copy followed by convolution. Besides the sharpness, mirror-links could also make the training converge faster.

The segmentation network produces $K+1$ labels for each cell in the 3D grid with K labels correspond to K points in that cell and one additional cell-level label. To obtain the ground truth labels for object parts at the cell-level, we take the majority label among labels of points in each cell. Cells with no point (and hence are filled with zeros) are labeled as "no label" and so are all the points within those cells. In testing, if there are less than or equal to K points in each cell, we use the corresponding K labels for each of them. Otherwise, the remaining points take the cell-level label.

3.4. Implementation details

3.4.1 Data augmentation

During training, before sampling to input grid, we augment the point cloud on-the-fly by randomly rotating the object along the up-axis and jittering the position of each points by a Gaussian noise with zero mean and 0.02 standard deviation. The sampling layer of our PointGrid also serves as an additional data augmentation.

3.4.2 Training

We set batch size as 32, batch normalization initial decay as 0.5, batch normalization decay clipping as 0.99. The weights for classification and segmentation losses are 0.2 and 0.8, respectively. Both losses are cross-entropy. We use Adam optimizer [26] with initial learning rate of 10^{-4}. Adam realizes the benefits of both AdaGrad [8] and RMSProp [12]. Instead of adapting the parameter learning rates based on the average first moment (the mean) as in RMSProp, Adam also makes use of the average of the second moments of the gradients (the uncentered variance). We implement our PointGrid using the public deep learning library TensorFlow [1] and train it using Nvidia Titan X for 8 hours for classification and 20 hours for segmentation (equivalent to 100 epochs). PointGrid consumes 4.0GB and 8.4GB of memory for classification and segmentation network, respectively. These figures are for grid size $N = 16$, $K = 4$ for each cell and batch size of 32.

Table 1: Object classification results on ModelNet40 [57].

Method	Input	Accuracy Overall	Accuracy Avg. Class
SPH [25]	mesh	–	68.2
3DShapeNets [57]	volume	84.7	77.3
VoxNet [35]	volume	85.9	83.0
Subvolume [39]	volume	89.2	86.0
VRN (simple) [6]	volume	91.3	–
VRN (ensemble) [6]	volume	95.5	–
LFD [57]	image	–	75.5
MVCNN [49]	image	–	90.1
FusionNet [53]	volume & image	90.8	–
Set-Conv [41]	point	90.0	–
PointNet [38]	point	89.2	86.2
PointNet++ [40]	point	91.9	–
Kd-Net [27]	point	91.8	–
O-CNN [55]	point	90.6	–
PointGrid (ours)	point	92.0	88.9

Table 2: Object classification results on ShapeNet-55 [7].

Method	Input	Accuracy Overall	Accuracy Avg. Class
PointNet [38]	point	83.2	78.2
PointGrid (ours)	point	86.1	80.5

Table 3: Accuracy of PointGrid's alternative structures on ModelNet40 [57].

Grid \ K	1	2	4	8	16
$4\times4\times4$	77.3	80.8	81.9	85.9	84.7
$8\times8\times8$	87.1	87.9	87.5	88.3	87.4
$16\times16\times16$	90.0	91.9	92.0	91.4	91.0
$32\times32\times32$	91.7	92.2	92.4	92.7	92.4

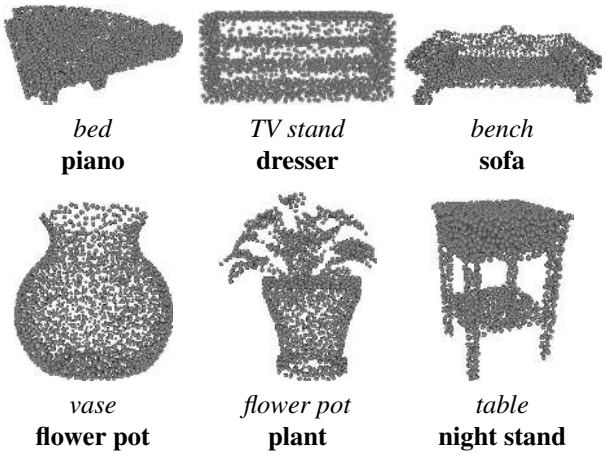


Figure 3: Some wrongly classified models. Predicted and actual labels are italicized and highlighted, respectively.

4. Experiments

We now discuss some experimental results of applying PointGrid to shape classification and object-part segmentation benchmark datasets. For classification, we evaluate PointGrid with various hyper-parameters such as the grid size (N) and the number of points per cell (K).

4.1. Shape Classification

Datasets: The ModelNet40 [57] benchmark contains 12,311 CAD models from 40 man-made object categories, which have been extensively used for 3D shape classification. The dataset is split into the training (9843 models) and the testing (2468) sets. ShapeNet-Core55 [7] benchmark contains a total of 51,190 3D models with 55 categories and 204 subcategories. The models are normalized to a unit length cube and have a consistent upright orientation. 70% of the dataset is used for training, 10% for validation, and 20% for testing.

Sampling point cloud: Given a triangular mesh model, 3D point cloud is computed as follows: firstly, a given number of triangles are sampled with the probability proportional to their surface areas. Then, for the sampled triangle a random point was taken by the following sampling equation.

$$(1-\sqrt{r_1})A+\sqrt{r_1}(1-r_2)B+r_2\sqrt{r_1}C \qquad (1)$$

where A, B, C are the coordinates of the triangle's vertices and r_1, r_2 are random real numbers with $r_1, r_2 \sim U(0,1)$.

The whole sampling procedure thus closely approximated uniform sampling of model surfaces. For each model, we uniformly sample 1024 points on the surface.

Results: Our PointGrid gets 92.0% and 86.1% overall accuracy on ModelNet40 and ShapeNet-Core55 datasets, respectively. Fig. 3 shows some wrongly classified models.

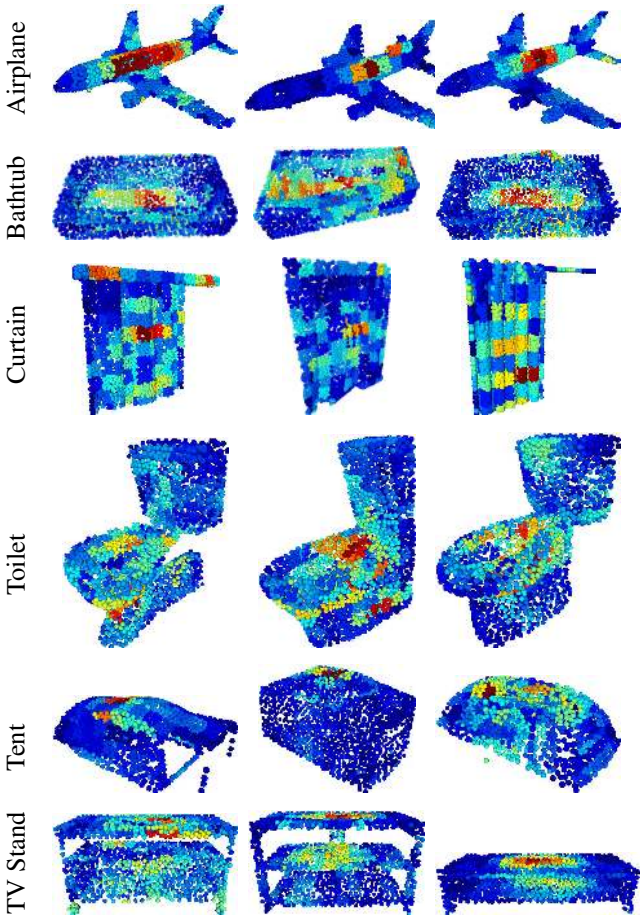


Figure 4: Visualization of object saliency. Magnitude of the gradient of the probability w.r.t. input grid is populated to points. Red indicates highly salient regions.

As we can see, there are still some ambiguities in categories whose appearances could be similar such as bench versus sofa and flower pot versus vase.

Comparison: In Table 1, we compare our method with a representative set of state of the art methods. In the category of "accuracy overall", our PointGrid performs better than the majority of other voxel-based approaches such as 3DShapeNet [57], VoxNet [35], Subvolume [39] and VRN (single) [6] while being worse than the VRN-ensemble [6] which involves an ensemble of 6 models each trained separately over the course of 6 days on NVidia Titan X. Comparing with methods applied to point cloud, despite of its simplicity, our network edges out Set-Conv [41], PointNet [38], PointNet++ [40], Kd-Net [27] and O-CNN [55]. One of the reason may be because our method better captures points' contextual neighborhood and hence learns better high-level features. There is still a small gap between our method and multi-view based method (MVCNN [49]) in the "average class accuracy" which may be due to the fact that MVCNN

Table 4: Object-part segmentation results on ShapeNet-part [62].

Method	# shapes		Wu	Yi	3D	Point-	Kd-Net	SpecCNN	O-CNN	PointGrid
	test	total	[56]	[62]	CNN	Net [38]	[27]	[63]	[55]	(ours)
mean IoU			–	81.4	79.4	83.7	77.2	84.7	85.9	86.4
airplane	341	2690	63.2	81.0	75.1	83.4	79.9	81.6	85.5	85.7
bag	14	76	–	78.4	72.8	78.7	71.2	81.7	87.1	82.5
cap	11	55	–	77.7	73.3	82.5	80.9	81.9	84.77	81.8
car	158	898	–	75.7	70.0	74.9	68.8	75.2	77.0	77.9
chair	704	3758	73.5	87.6	87.2	89.6	88.0	90.2	91.1	92.1
earphone	14	69	–	61.9	63.5	73.0	72.4	74.9	85.1	82.4
guitar	159	787	–	92.0	88.4	91.5	88.9	93.0	91.9	92.7
knife	80	392	–	85.4	79.6	85.9	86.4	86.1	87.4	85.8
lamp	286	1547	74.4	82.5	74.4	80.8	79.8	84.7	83.3	84.2
laptop	83	451	–	95.7	93.9	95.3	94.9	95.6	95.4	95.3
motor	51	202	–	70.6	58.7	65.2	55.8	66.7	56.9	65.2
mug	38	184	–	91.9	91.8	93.0	86.5	92.7	96.2	93.4
pistol	44	283	–	85.9	76.4	81.2	79.3	81.6	81.6	81.7
rocket	12	66	–	53.1	51.2	57.9	50.4	60.6	53.5	56.9
skateboard	31	152	–	69.8	65.3	72.8	71.1	82.9	74.1	73.5
table	848	5271	74.8	75.3	77.1	80.6	80.2	82.1	84.4	84.6

used a higher resolution to represent an object (80 views of 224×224 image versus $16 \times 16 \times 16 \times 4$ in our PointGrid).

Table 2 shows a side-by-side comparison between our PointGrid and PointNet [38]. Our method outperforms PointtNet [38] in both categories, 2.9% overall accuracy and 2.3% average-class accuracy.

Alternative network architecture: We conduct experiments with alternative PointGrid's architectures (on N and K) and report their overall accuracy for the object classification on ModelNet40 dataset [57]. According to Table 3, increasing the grid resolution (with extra memory and computational cost) increases the accuracy as it captures finer details. However, the improvement keeps decreasing each time we double the size of the grid. This might be due to the relatively lower sampling resolution of the 3D data in the experiment where only 1024 points are sampled for each model. Regarding to the number of points per cell K, again keep using more points is not always better. This may also be due to fact that with the lower sampling rate of the 3D data, the point cloud has already been well represented with K points per cell, and further increasing K could result in randomly duplicating data which could make it harder for the convolution to extract good features. We choose $N = 16$ and $K = 4$ for the best trade-off between accuracy and the memory/computation usage.

Visualization: Fig. 4 shows the magnitude of the gradient of the highest predicted probability with respect to the input cell with blue to red indicates low to high magnitude. We populate the gradient magnitude from each cell to all the underlying points. Loosely speaking, a large gradient's magnitude indicates that a small change of points within that cell results in a large change in its classification probability. Therefore, this figure could be thought as saliency map of the objects. It is interesting that to classify an object, PointGrid looks for some tube structure for airplane, curvy surface for curtain, roof for tent and flat horizontal surface for TV stand.

4.2. Object-part Segmentation

Dataset: We evaluate our architecture for object-part segmentation on ShapeNet-part dataset from [62], which augments a subset of the ShapeNet models with semantic part annotations. It contains 16,881 shapes represented as separate point clouds from 16 categories with per point annotation (with 2 to 6 parts per category and 50 parts in total). We use the same training/testing split provided by [38, 63]. In this dataset, both the object categories and the object parts within the categories are highly imbalanced, which poses a challenge to all methods including ours.

Comparison: Fig. 5 displays some examples of segmentation results of our PointGrid compared with those of PointNet [38]. As we can see, our results are visually better in most cases. For example, our PointGrid can separate out the wheels from the body of a motorcycle while PointNet cannot. Similar observations can be made for other models such as the pistol, skateboard and cap.

Table 4 numerically compares the segmentation performance of our PointGrid against other deep learning approaches. Evaluation metric is per-category and mean IoU on points. In the category of "mean IoU", our PointGrid

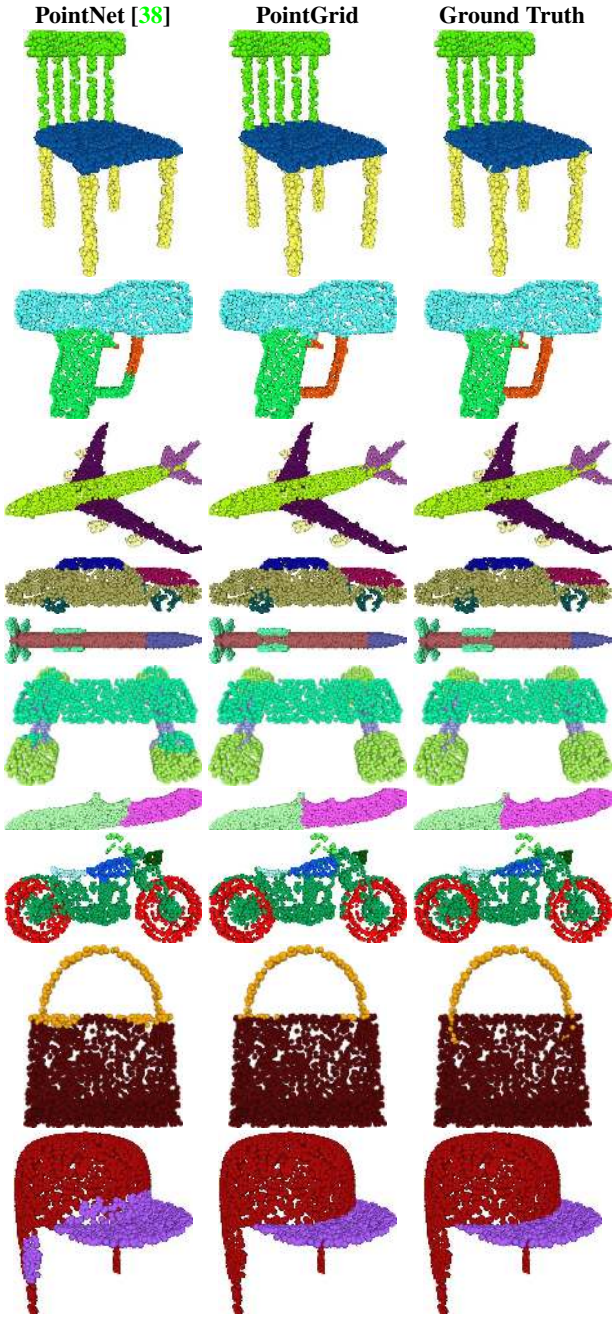


Figure 5: Comparison between PointGrid and PointNet on object-part segmentation. This result is based on grid size $16 \times 16 \times 16$ and $K = 4$.

achieves the best accuracy. In individual categories, we rank the best in airplane, car, chair and table; the second best in bag, earphone, guitar, lamp, mug and pistol; the third in cap, motor, rocket and skateboard. Our method places the fourth in laptop, and the fifth in knife. As we can see, our method performs better when there is more data in the category such as airplane, chair, table, etc. Note that we only use grid size of $16 \times 16 \times 16$ where all the other volumetric methods use at least $32 \times 32 \times 32$ grid. Moreover, we do not have any post-processing step such as boundary refinement by Conditional Random Field as is done in O-CNN [55].

Table 5: Average testing time on ModelNet40 [57].

	Classification	**Segmentation**
PointNet [38]	8.98ms	28.37ms
3D CNN (32^3)	27.50ms	64.32ms
3D CNN (64^3)	49.12ms	136.54ms
PointGrid (16^3)	14.91ms	48.94ms

Time complexity: Table 5 shows average inference time of our method on both classification and segmentation tasks (We test our method on individual data sequentially). For comparison, we run the authors' code of PointNet [38]. Since the 3DShapeNet [57] does not conduct segmentation, we extend the network in [57] to a fully convolutional network by omitting the fully connected layers and symmetrically adding the deconvolution layers. As we can see, PointGrid with grid resolution of 16^3 is faster than 3D CNN methods such as [57] with grid resolution of 32^3, and still outperforms it in the accuracy as shown in our paper. It is slower than PointNet but it performs better than PointNet in both classification and segmentation.

5. Conclusion

In this work we propose a new deep learning architecture PointGrid suitable for 3D visual recognition tasks such as 3D classification and semantic segmentation. PointGrid is a hybrid representation of point and grid that can better capture local geometric details while exhibits easy-to-learn regular structure. Experiments on widely used benchmark datasets [57, 7, 62] show that PointGrid compares favorably over existing deep learning methods [56, 62, 38, 27, 63, 55] on both classification and segmentation with significantly smaller memory footprint than other volumetric approaches.

Currently in the "point quantization" step of Section 3.1, if there are more than K points, we randomly sample K of them. Although it works well in our experiment, we would like to explore other advanced sampling techniques, e.g. furthest sampling [9], which may provide better representation of the local shape.

Acknowledgement

We would like to thank the authors of ModelNet40 [57] and ShapeNet [7, 62] datasets. We also appreciate Su et el. [38], Wang et al. [55] for making PointNet and O-CNN publicly available.

References

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org. 5

[2] A. Agathos, I. Pratikakis, S. Perantonis, N. Sapidis, and P. Azariadis. 3d mesh segmentation methodologies for cad applications. *Computer Aided Design and Applications*, 4(6):827–841, 2007. 2

[3] M. Attene, B. Falcidieno, and M. Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22:181–193, 2006. 2

[4] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522, 4 2002. 2

[5] W. Benjamin, A. W. Polk, S. Vishwanathan, and K. Ramani. Heat walk: Robust salient segmentation of non-rigid shapes. *Computer Graphics Forum*, 30(7):2097–2106, 2011. 2

[6] A. Brock, T. Lim, J. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. *Computing Research Repository - arXiv*, 2016. 1, 3, 5, 6

[7] A. X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, J. Xiao, L. Yi, and F. Yu. Shapenet: An information-rich 3d model repository. *Computing Research Repository - arXiv*, 2015. 5, 6, 8

[8] J. C. Duchi, E. Hazan, and Y. Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159, 2011. 5

[9] Y. Eldar, M. Lindenbaum, M. Porat, and Y. Y. Zeevi. The farthest point strategy for progressive image sampling. *IEEE Transactions on Image Processing*, 6(9):1305–1315, 9 1997. 8

[10] C. Farabet, C. Couprie, L. Najman, and Y. LeCun. Learning hierarchical features for scene labeling. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 35(8):1915–1929, 8 2013. 1

[11] M. Garland, A. Willmott, and P. S. Heckbert. Hierarchical face clustering on polygonal surfaces. In *Processing of the Symposium on Interactive 3D Graphics*, I3D '01, pages 49–58, 2001. 2

[12] K. S. Geoffrey Hinton, Nitish Srivastava. Lecture 6 - overview of mini-batch gradient descent. Computer Science lecture at University of Toronto. 5

[13] A. Golovinskiy and T. Funkhouser. Randomized cuts for 3d mesh analysis. *ACM Transactions on Graphics*, 27(5):145:1–145:12, 12 2008. 2

[14] A. Golovinskiy, V. G. Kim, and T. Funkhouser. Shape-based recognition of 3D point clouds in urban environments. *International Conference on Computer Vision*, 9 2009. 2

[15] K. Guo, D. Zou, and X. Chen. 3d mesh labeling via deep convolutional neural networks. *ACM Transactions on Graphics*, 35(1):3:1–3:12, 12 2015. 3

[16] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 6 2016. 1, 3

[17] M. Hilaga, Y. Shinagawa, T. Kohmura, and T. L. Kunii. Topology matching for fully automatic similarity estimation of 3d shapes. In *ACM Transactions on Graphics*, pages 203–212, 2001. 2

[18] S. Hong, J. Oh, H. Lee, and B. Han. Learning transferrable knowledge for semantic segmentation with deep convolutional neural network. *Computing Research Repository - arXiv*, 2015. 1

[19] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, 2015. 4

[20] A. Jagannathan and E. Miller. Three-dimensional surface mesh segmentation using curvedness-based region growing approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 29(12):2195–2204, 12 2007. 2

[21] A. E. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433–449, 5 1999. 2

[22] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri. 3d shape segmentation with projective convolutional networks. *Computing Research Repository - arXiv*, 2016. 3

[23] E. Kalogerakis, A. Hertzmann, and K. Singh. Learning 3d mesh segmentation and labeling. *ACM Transactions on Graphics*, 29(3), 2010. 2

[24] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954–961, 7 2003. 2

[25] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *Eurographics/ACM SIGGRAPH Symposium on Geometry Processing*, pages 156–164, 2003. 5

[26] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *IEEE International Conference for Learning Representations*, 2015. 5

[27] R. Klokov and V. S. Lempitsky. Escape from cells: Deep kd-networks for the recognition of 3d point cloud models. *Computing Research Repository - arXiv*, 2017. 1, 2, 3, 5, 6, 7, 8

[28] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012. 1, 4

[29] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin. Fast mesh segmentation using random walks. In *Processing of the ACM Symposium on Solid and Physical Modeling*, SPM '08, pages 183–191, 2008. 2

[30] G. Lavoue, F. Dupont, and A. Baskurt. A new cad mesh segmentation method based on curvature tensor analysis. *Computer-Aided Design*, 37(10):975–987, 2005. 2

[31] T. Le, G. Bui, and Y. Duan. A multi-view recurrent neural network for 3d mesh segmentation. *Computers & Graphics*, 66(Supplement C):103–112, 2017. 3

[32] Y. Li, S. Pirk, H. Su, C. R. Qi, and L. J. Guibas. Fpnn: Field probing neural networks for 3d data. In *IEEE International Conference on Neural Information Processing Systems*, pages 307–315, 2016. 1, 3

[33] R. Liu, H. Zhang, A. Shamir, and D. Cohen-Or. A part-aware surface metric for shape analysis. *Computer Graphics Forum*, 28(2):397–406, 2009. 2

[34] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *IEEE International Conference on Pattern Recognition*, 11 2015. 1

[35] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 922–928, 9 2015. 1, 3, 5, 6

[36] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *IEEE International Conference on Machine Learning*, pages 807–814, 2010. 4

[37] H. Noh, S. Hong, and B. Han. Learning deconvolution network for semantic segmentation. In *IEEE International Conference on Computer Vision*, 2015. 1

[38] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *Computing Research Repository - arXiv*, 2016. 1, 2, 3, 5, 6, 7, 8

[39] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. Guibas. Volumetric and multi-view CNNs for object classification on 3d data. In *IEEE International Conference on Computer Vision and Pattern Recognition*, 2016. 1, 3, 5, 6

[40] C. R. Qi, L. Yi, H. Su, and L. J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Computing Research Repository - arXiv*, 2017. 2, 3, 5, 6

[41] S. Ravanbakhsh, J. Schneider, and B. Poczos. Deep learning with sets and point clouds. *Computing Research Repository - arXiv*, 2016. 5, 6

[42] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2017. 1, 2, 3

[43] A. Shamir. A survey on mesh segmentation techniques. *Computer Graphics Forum*, 27(6):1539–1556, 9 2008. 2

[44] L. Shapira, A. Shamir, and D. Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249–259, 2008. 2

[45] A. Sharma, O. Tuzel, and D. W. Jacobs. Deep hierarchical parsing for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 530–538, 6 2015. 1

[46] J. Shi and J. Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, 8 2000. 2

[47] Z. Shu, C. Qi, S. Xin, C. Hu, L. Wang, Y. Zhang, and L. Liu. Unsupervised 3d shape segmentation and co-segmentation via deep learning. *Computer Aided Geometric Design*, 43:39–52, 2016. Geometric Modeling and Processing 2016. 2

[48] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *International Conference on Learning Representations*, 2014. 1

[49] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *IEEE International Conference on Computer Vision*, 2015. 3, 5, 6

[50] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 6 2015. 1, 3

[51] P. Theologou, I. Pratikakis, and T. Theoharis. A comprehensive overview of methodologies and performance evaluation frameworks in 3d mesh segmentation. *Computer Vision and Image Understanding*, 135:49–82, 2015. 2

[52] M. Vieira and K. Shimada. Surface mesh segmentation and smooth surface extraction through region growing. *Computer Aided Geometric Design*, 22:771–792, 2005. 2

[53] R. Z. Vishakh Hegde. Fusionnet: 3d object classification using multiple data representations. *Computing Research Repository - arXiv*, 2016. 5

[54] D. Z. Wang and I. Posner. Voting for voting in online point cloud object detection. In *Robotics: Science and Systems*, 2015. 1, 3

[55] P.-S. Wang, Y. Liu, Y.-X. Guo, C.-Y. Sun, and X. Tong. O-CNN: Octree-based convolutional neural networks for 3d shape analysis. *ACM Transactions on Graphics*, 36(4):72:1–72:11, 7 2017. 1, 2, 3, 5, 6, 7, 8

[56] Z. Wu, R. Shou, Y. Wang, and X. Liu. Interactive shape co-segmentation via label propagation. *Computers & Graphics*, 38:248–254, 2014. 2, 7, 8

[57] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *IEEE International Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 1, 3, 5, 6, 7, 8

[58] D. Xiao, H. Lin, C. Xian, and S. Gao. Cad mesh model segmentation by clustering. *Computers & Graphics*, 35(3):685–691, 2011. Shape Modeling International (SMI) Conference 2011. 2

[59] Z. Xie, K. Xu, W. Shan, L. Liu, Y. Xiong, and H. Huang. Projective feature learning for 3d shapes with multi-view depth images. *Computer Graphics Forum*, 34(7):1–11, 2015. 3

[60] H. Yamauchi, S. Lee, Y. Lee, Y. Ohtake, A. G. Belyaev, and H.-P. Seidel. Feature sensitive mesh segmentation with mean shift. In *In Processing of the International Conference on Shape Modeling and Applications*, pages 238–245, 2005. 2

[61] B. Yi, Z. Liu, J. Tan, F. Cheng, G. Duan, and L. Liu. Shape recognition of cad models via iterative slippage analysis. *Computer Aided Design*, 55(0):13–25, 2014. 2

[62] L. Yi, V. G. Kim, D. Ceylan, I.-C. Shen, M. Yan, H. Su, C. Lu, Q. Huang, A. Sheffer, and L. Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics*, 35(6):210:1–210:12, 11 2016. 2, 7, 8

[63] L. Yi, H. Su, X. Guo, and L. J. Guibas. SyncSpecCNN: Synchronized spectral CNN for 3d shape segmentation. *Computing Research Repository - arXiv*, 2016. 3, 7, 8

[64] E. Zhang, K. Mischaikow, and G. Turk. Feature-based surface parameterization and texture mapping. *ACM Transactions on Graphics*, 24(1):1–27, 1 2005. 2