Structured 2D Representation of 3D Data for Shape Processing

Kripasindhu Sarkar\textsuperscript{1,2}, Elizabeth Mathews\textsuperscript{1,2}, Didier Stricker\textsuperscript{1,2}
\{kripasindhu.sarkar, elizabeth.mathews, thekkekarottu\} didier.stricker@dfki.de
\textsuperscript{1} DFKI, Kaiserslautern \textsuperscript{2} Technische Universität Kaiserslautern

Abstract

We represent 3D shape by structured 2D representations of fixed length making it feasible to apply well investigated 2D convolutional neural networks (CNN) for both discriminative and geometric tasks on 3D shapes. We first provide a general introduction to such structured descriptors, analyze their different forms and show how a simple 2D CNN can be used to achieve good classification result. With a specialized classification network for images and our structured representation, we achieve the classification accuracy of 99.7\% in the ModelNet40 test set - improving the previous state-of-the-art by a large margin. We finally provide a novel framework for performing the geometric task of 3D segmentation using 2D CNNs and the structured representation - concluding the utility of such descriptors for both discriminative and geometric tasks.

1. Introduction

Convolutional Neural Network (CNN) has been a very powerful tool in solving problems in 2D domain [14, 30, 9, 5, 6, 24, 16, 8, 23, 15, 20]. Because of the presence of convolution operation (as convolutional layer), these networks require structured data as input - which maps to 2D images seamlessly. However, because of the unstructured and unordered nature of 3D data, application of such learning methods on 3D domain is not straightforward. Therefore, there have been various attempts to transcribe 3D data to a common parameterization - both in 3D and 2D.

In the first strategy, 3D structured descriptors such as voxel occupancy provide a structured grid for the application of 3D convolutions. However, because of the expensive nature of the operation, the memory consumption of 3D CNN increases ‘cubically’ w.r.t. the input resolution: limiting the existing networks to an input of size $32^3$ [19, 36, 32, 1]. In the second strategy and an attempt to avoid the memory expensive 3D convolutions, 3D shapes are converted to 2D grid like structure for the application of 2D CNNs [7, 3, 26, 57]. These existing methods design specific 2D descriptors suitable for the problem in hand, which are often not transferable to other tasks. As a result, structured 2D representation for 3D data is not standardized - the way voxel grid and 2D images are in the 3D and 2D domain respectively.

In this paper, we standardize structured 2D representation of 3D data and provide their general description. Based on their properties, we categorize them into different forms and analyze them. We provide a general network structure for performing discriminative tasks on such representation and show how it can be easily integrated to any well investigated 2D CNN designed for specific tasks. We verify this by performing classification of 3D shapes by both vanilla and specialized 2D CNN with the general structured representation as the input. Using a specialized 2D CNN for classification we achieve state-of-the-art accuracy in the competitive ModelNet40 benchmark [36] with all of the forms of the representations. Our classification result is summarized.

Figure 1: Evolution of classification methods and their performance on ModelNet40 benchmark [36]. The X-axis here shows a rough chronological order of the publications. Using our 10-Layered Slice descriptors and a 2D CNN for classification, we achieve 99.7\% accuracy in the ModelNet40 test set.
in Figure 1.

Solving geometric tasks such as 3D segmentation using a 2D descriptor is challenging, because a significant amount of information is lost with the reduction of a dimension. In this paper we propose a novel architecture for performing the geometric task of 3D segmentation with height-map based representation as input. Therefore, we show the utility of the structured 2D descriptors in terms of both discriminative and geometric tasks.

The descriptors analyzed in this paper does not need a pre-estimation of 3D mesh structure and topology, and can be computed directly on unordered point clouds. This is in contrast to shape-aware parameterizations that require the knowledge of 3D mesh topology, e.g. mesh quadrangulation, intrinsic shape description in eigenspace of the mesh Laplacian etc [18, 27]. Structured 2D representation is suited for learning shape features in a large dataset, and therefore is ideal for the current data driven approach of neural networks. As argued before, it is comparable to the standard voxel grid descriptor, but in 2D. It enables all the advantage of 2D CNNs, such as ease of performing convolution, initialization of network weights pretrained on millions of data samples, efficiency of memory etc in the context of 3D data. It also makes it possible to use well designed core 2D networks such as GoogLeNet [33], ResNet [9] for shape analysis. Our contributions are the following:

- We generalize structured 2D representations of 3D data and their usage with 2D CNN for discriminative tasks.
- Using the structured 2D representations as input and a CNN framework for classification, we achieve 99.7% classification accuracy in the ModelNet40 [36] test set and improve the previous state-of-the-art by a large margin.
- We then propose a novel network for performing geometric tasks on the 2D descriptors and solve the problem of 3D part segmentation and semantic segmentation.

With the publication, we will make different structured descriptors of ModelNet40 (from different orientations) and the implementation of our network public.

The following section describes the related literature. Section 3 describes the structured 2D representation and its different forms. Section 4 introduces general classification methods on the representation followed by the details of the specialized network, that jointly classifies and detects the orientation. We follow it by introducing our novel framework for segmentation in Section 5. We finally present our result of the experiments in Section 6.

2. Related Work

3D voxel grid for 3D convolution  In these methods, 3D shape is represented as a binary occupancy grid in a 3D voxel grid on which 3D CNN is applied. Wu et al.[36] uses deep 3D CNN for voxelized shapes and provides the popular classification benchmark dataset of ModelNet40 and ModelNet10. This work is quickly followed by network design that take ideas from popular 2D CNNs giving a big boost in performance over the baseline [19, 1]. [22, 28] design special CNNs optimized for the task of 3D classification. However, because of the fundamental problem of memory overhead associated with 3D networks, the input size was restricted to $32^3$, making them the least accurate methods for both discriminative and geometric tasks. In contrast to voxel grid, we use structured 2D descriptors and use 2D CNN and perform better in both classification and segmentation.

Rendered images as descriptor  These methods take virtual snapshots of the shape as input descriptors and perform the task of classification using 2D CNN architecture. Their contributions are novel feature descriptors based on rendering [29, 10, 32] and specialized network design for the purpose of classification [34, 12, 4]. The specialized CNN used for classification in this paper is inspired from [12] where classification and orientation estimation is jointly performed to increase the classification performance. Even though rendered images by definition structured 2D descriptors, they do not provide any direct geometric information. Because of this reason they are not considered a part of the representation in this paper. With the same network architecture all forms of our representation perform significantly better than the rendered images.

2D representations for 3D tasks  Detection: These methods project the point-cloud collected from 3D sensors such as LIDAR onto a plane and discretize them to a 2D grid for 2D convolution for 3D object detection [3, 37, 17]. The projection on the ground plane, which is often referred as 'Bird Eye View', is augmented with other information and finally fed to a network designed for 3D detection. Here the 3D data is assumed to be sparse along the Z direction - across which convolution is performed.

Classification: Gomez-Donoso et al. [7] represents shape by '2D slices' - the binary occupancy information along the cross section of the shape at a fixed height. Sarkar et al. [26] on the other hand represents shape by height-map at multiple layers from a 2D grid. Both of them combines descriptors from different views by a MVCNN [32] like architecture for classification.

We standardize structured 2D representation of 3D data with the ideas spanning through all these works. We provide a general description of such representation, that can be used by any 2D CNN designed for images classification.
Specialized networks for 3D  Recently, there has been a serious effort to have alternative ways of applying CNNs in 3D data such as OctNet [25] and PointNet [21]. OctNet uses a compact version of voxel based representation where only the occupied grids are stored in an octree instead of the entire voxel grid. PointNet [21] takes unstructured 3D points as input and gets a global feature by using max pool as a symmetrical function on the output of multi-layer perceptron on individual points. Our method is conceptually different, as it respects the actual spatial ordering of points in the 3D space.

3. Structured 2D representation

In this section, we provide a general description of the structured 2D representation of 3D data. The ideas of such descriptors are taken from the previous literature in 3D computer vision [7, 3, 26, 37]. We start by providing the notation which is followed throughout the paper.

Notation  Multidimensional arrays are denoted by bold letters (e.g. \(x, y\)) and scalars are denoted by non-bold letters (e.g. \(x, \alpha\)). For a given integer \(n \in \mathbb{N}\), we define \([n] := \{1, \ldots, n\}\). Let \(A\) be a multidimensional array. \(A_{i,j,k,\ldots}\) denotes the element of \(A\) at the index \(i, j, k, \ldots\). Using : as index means to select all indices along that axis. Depending on the context, we use the subscript notation for denoting the sample number as well.

Formulation  Let \(P \in \mathbb{R}^{3 \times N}\) be a 3D point-cloud comprising \(N\) points, where the i-th column denotes an individual point. The point-cloud is obtained either directly from the 3D scanners such as Kinect, Intel RealSense, LIDAR, etc., or is sampled from a shape. Unlike pixel arrays in images, point-cloud consists of points without any order making them un-ordered and unstructured. This makes it difficult to use them as input to standard machine learning algorithms - such as Convolutional Neural Networks. Given such unstructured point-cloud \(P\), we convert them into a structured 2D representation \(x\) of dimension \(H \times W \times c\). This descriptor is both structured and of fixed-length, which enables the application of standard machine learning algorithm.

3.1. Learning methods on structured input

Given data-samples from structured representation \(X \subset \mathbb{R}^{H \times W \times c}\), we search a function \(f : X \mapsto Y\) mapping an input \(x \in X\) to a task specific output \(y \in Y\). The function \(f\) is often known as the ‘model’ or ‘score function’. Some examples of such models are linear-model, multi-layer perceptron and convolutional neural network (CNN). In the case of CNN, 2D convolutional filters are applied across the first two spatial dimensions of the input (of size \((H, W)\)), while 3rd dimension is taken as the depth of the convolutional filter. Thus the structured 2D representation is encoded as \(c\)-channel (or \(c\)-layered) features of \(H \times W\) spatial dimensions. We use the term ‘channel’ and ‘layer’ interchangeably in this paper to denote the depth axis. Therefore in this notation, \(k\)-th layer of the feature \(x\) is \(x_{c\ldots k}\).

3.2. Forms of structured 2D representations

As a first step of 2D discritization of 3D shapes, a discrete 2D grid of spatial dimension \(H \times W\) is placed on the 3D data. For each bin of the 2D grid \(c\) values are sampled according to a criteria. Inspired by the previous literature, we mention here different strategies of sampling.

3.2.1 Layered height-maps

In this strategy for each bin of the 2D grid, \(c\) physical height values are sampled from all the points falling into the bin. i.e. \(x_{i,j,c}\) represents a height value of a sampled-point from the grid plane falling at the bin \((i, j)\). The \(c\) height values are chosen as percentile statistics to uniformly distribute height information in all the layers. i.e. \(k^{th}\) layer is computed as \(\frac{k-1}{c} \times 100^{th}\) percentile of the height values of the points falling in a bin, which evenly samples the height values w.r.t. number of points for each layers (1st layer \(\rightarrow\) minimum height, \(c^{th}\) layer \(\rightarrow\) maximum height). An approximation of this strategy can be achieved by dividing points equally into \(c\) slices and computing a height value for each
slice [3]. We use the percentile method when we compute this descriptor and address it as ‘multi-layered height-map (MLH)’ or ‘layered height-map’ in this paper.

3.2.2 Occupancy slices

In this strategy, occupancy slices are computed at predetermined height values. That is for each bin of the 2D grid, at c fixed height values across the first two dimensions, slices are made of a specific thickness t. Binary occupancy information of the points in the voxel comprising of the 2D bin (the first two dimension) and the slice (the 3rd dimension) are taken as the feature. The slices are considered at heights equidistant to each other to evenly cover the shape cross-section. We address this descriptor as ‘Slice’ or ‘Occupancy slices’ in this paper.

3.2.3 Binary occupancy volume

In this strategy, the entire \( H \times W \times c \) dimension tensor represents binary occupancy information of the 3D data. That is a physical dimension \( L \times B \times D \) is defined for the scene and a 3D grid of shape \( H \times W \times c \) is placed on the data. The points inside this cubical space is then discretized with a resolution of \( \frac{L}{c} \times \frac{B}{c} \times \frac{D}{c} \) as occupancy information. Note that, this is same as the traditional voxel occupancy descriptors. The important differences are a) the resolution \( \frac{L}{c} \) along the depth dimension is often different to that of the spatial dimension \( \frac{B}{c} \) b) 2D convolution operation is applied along the special dimension instead of expensive 3D convolution along all three axes.

We point out that the binary occupancy volume descriptor reduces to occupancy slices when \( t = \frac{L}{c} \), and slices taken at \( c \) even intervals along the depth dimension. Because of their similarity, we chose occupancy slices descriptors to represent both the categories. Figure 2 provides a visualization of the different forms of the aforementioned descriptors.

3.3. Orientation views

As explained before, structured 2D representation requires placement of 2D grid on the point cloud as a first step. Therefore, the feature descriptors are dependent on orientation of the grid; where orientation is defined as the direction of the plane of the grid. A good system should be able to merge the descriptors of an instance coming from different orientations. This can be treated analogous to the ‘camera views’ for generating rendered images of 3D shapes and using them as feature descriptors [29, 10, 32, 34, 12]. We use the geometric structured representation and exploit the careful design choices of the image based network for merging orientation to perform the task of 3D classification.

4. Classification

4.1. Vanilla classification system from a single orientation

In the simple case, structured 2D representations of mean-centered and normalized shapes can be computed w.r.t. to a canonical axis, as its orientation and a classification based learning method can be applied as described in Section 3.1. Here each descriptor \( x \in \mathbb{R}^{H \times W \times c} \) is assigned a class label \( c \in [n] \), where \( n \) is the total number of classes. The network \( f \), with a normalization such as softmax at the last layer, outputs \( n \) probability estimation \( y \in [0, 1]^n \) for each class. The true distribution \( *y \) is then the one-hot encoding for the class. i.e. its k-th element,

\[
y_k = \begin{cases} 1 & k = c, \ 0 & \text{otherwise} \end{cases}
\]

The dissimilarity between the two distribution is measured by a loss function \( \mathcal{L} : Y \times \hat{Y} \rightarrow \mathbb{R} \). A commonly used function for classification is the cross-entropy between two distribution. It is given by

\[
\mathcal{L}(y, \hat{y}) = - \log \hat{y}_c
\]

Therefore, we can use a simple feed forward 2D CNNs like VGG, ResNet, etc. whose parameters are optimized by a classification loss. This enables classification from the descriptors computed from a single orientation. To increase the performance of classification, we can either perform orientation augmentation (or ensemble of classifiers) or fine grained analysis of orientations clusters - inspired from the networks that uses 2D images. The main idea is to get \( m \) different descriptors of an instance of 3D model from \( m \) different orientation and fuse them in a careful manner for the final classification. The following subsection describes such method in detail.

4.2. RotationNet for 3D descriptors

Inspired by the network in [12], we jointly perform classification and orientation estimation for better aggregation of feature descriptors from different orientations. It is shown that when synthetic images of 3D models are used, the classification performance improves by joint estimation of orientation and categories [28].

For each instance of the 3D shape we compute different descriptors \( \{x_i\}_{i=1}^m \) from \( m \) pre-defined orientations. A orientation variable \( o_i \in [m] \) is assigned to \( x_i \) such that, \( o_i = j \) when \( x_i \) is computed from \( j \)-th orientation. A new class ‘incorrect view’ with class id \( n + 1 \) is introduced in addition to the \( n \) different classes for classification. The model consists of a base multi-layered network followed by the concatenation of \( m \) softmax layers which outputs the class probabilities \( y \in [0, 1]^{m \times (n+1)} \) w.r.t different views. For the input \( x_i \), the true distribution \( *y \) is defined such that its (j,k)-th element \( y_{j,k} \) is given by
3.2.2) and binary occupancy volume (Section 3.2.3) are occupancy based descriptors like voxel grids, 3D geometric tasks like segmentation can be solved in a similar way, as it is done for voxel grids using a 3D CNN. Performing such tasks using a 2D CNN is not straightforward.

In contrast, multi-layered height-map (Section 3.2.1) descriptors have rich information along the direction perpendicular to the plane of the grid as it directly stores the height information - making them a good choice for performing geometric tasks. In this section we introduce our novel method, that uses 2D multi-layered height-map as input and solves the problem of point-cloud segmentation.

5.1. Single view network

Input representation In point-cloud segmentation each point $P_i$ contains a segmentation label $l \in [n]$, where $n$ is the total number of class labels. We convert such labeled point-cloud to 2D MLH descriptor $x \in \mathbb{R}^{H \times W \times c}$, where $x_{i,j,c}$ denotes the displacement value from the grid. We also prepare a 2D grid for the labels $l \in [n]^{H \times W \times c}$ such that $l_{i,j,c}$ has the same label as the point that was considered for the displacement value at $x_{i,j,c}$.

Network formulation Given a set input descriptors in the form of $x$ and the label maps in the form of $l$, we solve 2D segmentation task for each layer. That is, we solve $c$ segmentation task using a 2D CNN with $x$ as input and $l_{i,:,c}$ as output labels, where $k \in [c]$.

The 2D segmentation network has two parts: 1) a base network and 2) $c$ classification heads responsible for the final pixelwise classification for all the $c$ layers. In our implementation we chose the final 3 fully convolutional layers of the segmentation network as classification head. The final loss is the summation of all the losses coming from the classification heads. We finally backpropagate the entire network with the gradient computed from the final loss. The network architecture is summarized in Figure 4.

Note that, the 2D CNN for segmentation can be chosen to be any of the well investigated architectures such as FCN, UNet, PSPNet, etc [16, 40, 2]. The only difference is, that we have $c$ different final layers for classification after the base network.
5.2. Multi-view architecture

The MLH descriptors are lossy in nature. Surfaces that are approximately perpendicular to the 2D grid of the descriptors are represented by \( c \) values, which results to a large loss of geometrical information. This causes large empty areas in the computed label voxel grid \( V \), which leads to unlabelled points of the input cloud. To alleviate this issue, we compute the segmentation labels for 3 different orientations corresponding to the 3 axes of the input shape. We use the intuition, that a surface cannot be perpendicular to all the 3 axes of the input shape. We use the simple network of VGG (VGG16 with batch normalization) [30] as the base network for this task and show our result in Table 1. We use the ImageNet pretrained weights for initialization in all our experiments to take advantage of 2D CNNs trained on millions of samples. In the first convolutional layer where the depth of the filter doesn’t match \( c \) instead of 3, we repeat the weights channel-wise.

6. Experiments

In this section we provide our experimental evaluation of the structured 2D descriptors for the task of classification and segmentation.

6.1. 3D Shape Classification

Dataset and general settings We use the popular dataset of ModelNet40 [36] for evaluating our shape classification results. It contains approximately 12k shapes from 40 categories and comes with its own training and testing splits of around 10k and 2.5k shapes. We first densely sample points form each shape and compute multi-layered height-map (MLH) and occupancy slice feature descriptors of dimension \( 256 \times 256 \times c \). As found by the experiments (Section 6.1.3), we use \( c = 5 \) for most of the cases. We use the ImageNet pretrained weights for initialization in all our experiments to take advantage of 2D CNNs trained on millions of samples. In the first convolutional layer where the depth of the filter doesn’t match \( c \) instead of 3, we repeat the weights channel-wise.

| Descriptor     | Accuracy |
|----------------|----------|
| MLH 5 layer    | 91.05    |
| Slice 5 layer  | 88.21    |

Table 1: Classification accuracy on ModelNet40 with a single orientation.

6.1.1 Vanilla CNN for classification

We solve the problem of classification by a single vanilla CNN and a cross-entropy loss with the descriptors from a single orientation. We use the simple network of VGG (VGG16 with batch normalization) [30] as the base network for this task and show our result in Table 1. As seen, even with a single orientation and without performing any data augmentation and fine grained analysis, our result with MLH descriptors comes out to be better than the many multi-view baselines, including MVCNN [32]. This shows the strength and advantage of using structured 2D descriptor and 2D CNN for 3D classification.
| Method                    | Accuracy | Views / Test instances |
|---------------------------|----------|------------------------|
| PANORAMA NN [29]          | 91.12    | 1                      |
| Dominant Set Clustering [31] | 93.80    | 12                     |
| Pairwise [10]             | 90.70    | 12                     |
| MVCNN 12-v [32]           | 89.90    | 12                     |
| MVCNN 80-v [32]           | 90.10    | 80                     |
| Kd-Net depth 10 [13]      | 90.60    | 10                     |
| Kd-Net depth 15 [13]      | 91.80    | 10                     |
| MVCNN- MultiRes [22]      | 91.40    | 20                     |
| VRN [1]                   | 91.33    | 24                     |
| Voxception [1]            | 90.56    | 24                     |
| Pointnet [21]             | 89.20    | 1                      |
| VoxNet [19]               | 83.00    | 12                     |
| MLH-MV [26]               | 93.11    | 3                      |
| RotNet-orig 12-v [12]     | 90.08    | 12                     |
| RotNet-orig 20-v [12]     | 97.37    | 20                     |
| GVCNN [4]                 | 93.10    | 8                      |
| MHBN [39]                 | 94.70    | 6                      |
| VGG + MLH 1-v             | 91.05    | 1                      |
| Our RotNet + MLH-5L       | 99.56    | 12                     |
| Our RotNet + Slice-5L     | 99.51    | 12                     |
| Our RotNet + Slice-10L    | 99.76    | 12                     |

Table 2: Comparison of classification accuracy on ModelNet40 test split.

6.1.2 RotationNet for 3D descriptors

In this section we present our result of classification from the network described in Section 4.2. As discussed, for each mean-centered instance we compute 12 descriptors from the grid rotated along the Z axis. Using VGG as our base network we iteratively optimize the network and the orientation candidates based on Equation 5 and train till the convergence. We use a standard SGD with the initial learning rate of 0.01 and momentum of 0.9.

We use both MLH and Slice descriptors with different number of layers in this experiment and present our result in Table 2. As seen, our RotationNet with both MLH and Slice descriptors improve the state-of-the-art by a high margin. It is to be noted, that occupancy slice descriptor (for 5 layered feature) is around 1/10-th of the size of MLH. Even with such less amount of information it achieves comparable result with MLH. This shows the superiority of slice descriptor on classification tasks.

6.1.3 Analysis of structured 2D representation

We perform experiments to analyze the optimal number of channels/layers for the structured 2D representations. We keep the same experimental setup of 12 view RotationNet, but change the number of layers for both MLH and Slice descriptors. For 1 layer descriptors, we took the minimum height for MLH, and slice occupancy at the mid-section of the shape for Slice descriptor. The result is shown in Figure 5.

We find the single layered structured descriptor to perform extremely poorly in comparison to their multi-layered versions. The result improves quickly with the addition of more layers and saturates after 5 with an accuracy of around 99.5%. Adding more layers in MLH does not increase the accuracy further. In fact, around 10 layer MLH we see a drop in the accuracy. Note that, due to the presence of the floating point height values, 5 layer MLH descriptor already consumes a significant amount of memory. With occupancy slice the accuracy increases upto 99.7% with 10 layers and does not improve further.

6.2 3D Segmentation

6.2.1 Experimental setup

Dataset and feature computation We use the 3D part segmentation dataset - ShapeNet parts - provided by [38] to evaluate our segmentation method. This dataset is a subset of ShapeNetCore and its segmentation labels are verified manually by experts. It also comes with its own training and testing split. We use 4 categories with the most training instances namely - Table, Chair, Airplane, and Lamp - for our evaluation as other categories do not have enough instances for training a deep neural network; and to compare with [35] whose results are only available for these categories. Note that, we deal with each category separately with different network. It can be argued, that our performance could improve further (and lack of training data for other categories be resolved) if we train our network across different categories with an external input of the category information in the final classification stage. Our intention here is not to push for the best result, but to validate our framework for 3D point segmentation by 2D descriptors.

We use our multi-view architecture for segmentation, and therefore, we compute features from the three axes as orientations (Section 5.2). Because of the low point density...
Figure 6: Qualitative result of our 3D segmentation on the ShapeNet-parts test split.

in the shapes of the segmentation dataset (≈ 3000), we compute MLH features and label-map of dimension $64 \times 64 \times 2$. We find, that 2 layers are sufficient for the semantic information of the shape as it captures the shape outline from outside. At empty regions in the label map, we put an additional 'invalid' label. Therefore, during the optimization we classify for $n + 1$ classes, where $n$ is the original number of classes.

2D segmentation network  As discussed, our segmentation framework is not tied to any particular type of 2D segmentation networks. We use the simple model of Fully Convolutional Network (FCN8s) [16] as our base CNN. For the classification heads we choose the final 3 convolutional layers of the network. We added the loss coming from the two classification heads and optimized the network through the backpropagation based on this resultant loss (Figure 4). For training, we use the pretrained VGG weights for initialization and a standard SGD optimizer with momentum with the initial learning rate as 1.0e-10. We perform segmentation of each category by a different network and show our qualitative result in Figure 6. In Figure 7 we show the 2D segmentation result of different layers and views of a Table.

6.2.2 Analysis

We use the point IoU between the ground truth and predicted labels as our evaluation metric. The mean IoU of a category is taken as the mean IoU of all instances of the category. We then compare our result with the method of Wu et al.[35], PointNet [21], a baseline 3D CNN for segmentation used in [21] and present our results in Table 3. It is seen that our result is quantitatively comparable to the existing state-of-the-art methods.

Our aim here is to provide an experimental verification of our segmentation framework instead of network engineering for the best result. The result can be improved by the following:

1. Use of latest 2D segmentation network such as PSPNet [40], DeepLab [2], etc instead of the basic FCNs to improve the 2D segmentation result.

2. Use of 3D meshes of ShapeNet-parts [11] to get a denser point-cloud to increase the input resolution of the MLH features (without introducing large amount of empty regions).

7. Conclusion

In this paper we provided a general introduction to structured 2D representation of 3D data and analyzed its various forms. We showed how a simple vanilla CNN like VGG can be use with such 2D descriptors to achieve good classification result. We then used a specialized 2D CNN to aggregate feature descriptors from different orientation to achieve 99.7% accuracy in the ModelNet test state improving the previous state-of-the-art accuracy by a large margin. Finally, we provided our novel framework for performing the geometric task of 3D segmentation using 2D networks and structured 2D representation. Therefore, we provided general framework for the 2D descriptors for solving both discriminative and geometric task. We concluded, that occupancy-slice descriptor provides an excellent choice for shape classification because of its compactness. Where as multi-layered height-map provides a good choice for 2D segmentation because of its richness of information. We hope this work will make structured 2D representation more standard for 3D data processing in the future; similar to that of voxel occupancy grid.

|                | Table | Chair | Airplane | Lamp |
|----------------|-------|-------|----------|------|
| Wu et al. [35] | 74.8  | 73.5  | 63.2     | 74.4 |
| PointNet [21]  | 80.6  | 89.6  | 83.4     | 80.8 |
| 3D CNN [21]    | 77.1  | 87.2  | 75.1     | 74.4 |
| **Ours**       | 79.2  | 82.0  | 72.1     | 74.5 |

Table 3: Quantitative result of 3D Segmentation on ShapeNet-part in terms of mean point IoU %.

Figure 7: Example of segmented (Left) and ground truth (Right) label-map of a Table. Layers are shown in different rows, and the orientations are shown in different columns.
References

[1] A. Brock, T. Lim, J. M. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. CoRR, abs/1608.04236, 2016. 1, 2, 7

[2] L.-C. Chen, G. Papandreou, I. Kokkinos, K. Murphy, and A. L. Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. arXiv:1606.00915, 2016. 5, 8

[3] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia. Multi-view 3d object detection network for autonomous driving. In IEEE CVPR, 2017. 1, 2, 3

[4] Y. Feng, Z. Zhang, X. Zhao, R. Ji, and Y. Gao. Gvnn: Group-view convolutional neural networks for 3d shape recognition. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018. 1, 2, 7

[5] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2014. 1

[6] R. B. Girshick. Fast R-CNN. CoRR, abs/1504.08083, 2015. 1

[7] F. Gomez-Donoso, A. Garcia-Garcia, J. Garcia-Rodriguez, S. Orts-Escolano, and M. Cazorla. Lonchanet: A sliced-based cnn architecture for real-time 3d object recognition. In 2017 International Joint Conference on Neural Networks (IJCNN), pages 412–418, May 2017. 1, 2, 3

[8] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In Computer Vision (ICCV), 2017 IEEE International Conference on, pages 2980–2988. IEEE, 2017. 1

[9] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. CoRR, abs/1512.03385, 2015. 1, 2

[10] E. Johns, S. Leutenegger, and A. J. Davison. Pairwise decomposition of image sequences for active multi-view recognition. In Computer Vision and Pattern Recognition (CVPR), 2016 IEEE Conference on, pages 3813–3822. IEEE, 2016. 2, 4, 7

[11] E. Kalogerakis, M. Averkiou, S. Maji, and S. Chaudhuri. 3D shape segmentation with projective convolutional networks. In Proc. IEEE Computer Vision and Pattern Recognition (CVPR), 2017. 8

[12] A. Kanezaki, Y. Matsushita, and Y. Nishida. Rotationnet: Fast and furious: Real-time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 3569–3577, 2018. 2

[13] J. Masci, D. Boscaini, M. M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In IEEE Workshop on 3D Representation and Recognition (3DRR), pages 37–45, 2015. 2

[14] D. Maturana and S. Scherer. VoxNet: A 3D Convolutional Neural Network for Real-Time Object Recognition. In IROS, 2015. 1, 2, 7

[15] D. Pathak, P. Krähenbühl, J. Donahue, T. Darrell, and A. Efros. Context encoders: Feature learning by inpainting. 2016. 1

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. arXiv preprint arXiv:1612.00593, 2016. 1, 3, 7, 8

[17] C. R. Qi, H. Su, M. Nießner, A. Dai, M. Yan, and L. J. Guibas. Volumetric and multi-view cnns for object classification on 3d data. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 5648–5656, 2016. 1, 2, 5, 7

[18] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint arXiv:1511.06434, 2015. 1

[19] S. Ren, K. He, R. B. Girshick, and J. Sun. Faster R-CNN: towards real-time object detection with region proposal networks. CoRR, abs/1506.01497, 2015. 1

[20] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning 3d deep representations at high resolutions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017. 3

[21] K. Sarkar, B. Hampiholi, K. Varanasi, and D. Stricker. Learning 3d shapes as multi-layered height-maps using 2d convolutional networks. In The European Conference on Computer Vision (ECCV), September 2018. 1, 2, 3, 5, 7

[22] K. Sarkar, K. Varanasi, and D. Stricker. 3d shape processing by convolutional denoising autoencoders on local patches. In 2018 IEEE Winter Conference on Applications of Computer Vision, WACV 2018, Lake Tahoe, NV, USA, March 12-15, 2018, pages 1925–1934, 2018. 2

[23] N. Sedaghat, M. Zolfaghari, E. Amiri, and T. Brox. Orientation-boosted voxel nets for 3d object recognition. arXiv preprint arXiv:1604.03351, 2016. 2, 4

[24] K. Sifakis, T. Theoharis, and I. Pratikakis. Exploiting the PANORAMA Representation for Convolutional Neural Network Classification and Retrieval. In I. Pratikakis, F. Dupont, and M. Ovsjanikov, editors, Eurographics Workshop on 3D Object Retrieval. The Eurographics Association, 2017. 2, 4, 5, 7

[25] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. CoRR, abs/1409.1556, 2014. 1, 6
[31] A. A. Soltani, H. Huang, J. Wu, T. D. Kulkarni, and J. B. Tenenbaum. Synthesizing 3d shapes via modeling multi-view depth maps and silhouettes with deep generative networks. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1511–1519, 2017.

[32] H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In Proc. ICCV, 2015.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. Going deeper with convolutions. In Computer Vision and Pattern Recognition (CVPR), 2015.

[34] C. Wang, M. Pelillo, and K. Siddiqi. Dominant set clustering and pooling for multi-view 3d object recognition. In Proceedings of British Machine Vision Conference (BMVC), 2017.

[35] Z. Wu, R. Shou, Y. Wang, and X. Liu. Interactive shape cosegmentation via label propagation. Computers & Graphics, 38:248–254, 2014.

[36] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1912–1920, 2015.

[37] B. Yang, W. Luo, and R. Urtasun. Pixor: Real-time 3d object detection from point clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7652–7660, 2018.

[38] 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. SIGGRAPH Asia, 2016.

[39] T. Yu, J. Meng, and J. Yuan. Multi-view harmonized bilinear network for 3d object recognition. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 186–194, 2018.

[40] H. Zhao, J. Shi, X. Qi, X. Wang, and J. Jia. Pyramid scene parsing network. In The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), July 2017.