3D Object Classification via Spherical Projections

Zhangjie Cao
School of Software
Tsinghua University, China
caozhangjie14@gmail.com

Qixing Huang
Department of Computer Science
University of Texas at Austin, USA
huangqx@cs.utexas.edu

Karthik Ramani
School of Mechanical Engineering
Purdue University, USA
ramani@purdue.edu

Abstract

In this paper, we introduce a new method for classifying 3D objects. Our main idea is to project a 3D object onto a spherical domain centered around its barycenter and develop neural network to classify the spherical projection. We introduce two complementary projections. The first captures depth variations of a 3D object, and the second captures contour-information viewed from different angles. Spherical projections combine key advantages of two main-stream 3D classification methods: image-based and 3D-based. Specifically, spherical projections are locally planar, allowing us to use massive image datasets (e.g, ImageNet) for pre-training. Also spherical projections are similar to voxel-based methods, as they encode complete information of a 3D object in a single neural network capturing dependencies across different views. Our novel network design can fully utilize these advantages. Experimental results on ModelNet40 and ShapeNetCore show that our method is superior to prior methods.

1. Introduction

We perceive our physical world via different modalities (e.g., audio/text/images/videos/3D models). Compared to other modalities, 3D models provide the most accurate encoding of physical objects. Developing algorithms to understand and process 3D geometry is vital to automatic understanding of our physical environment. Algorithms for 3D data analysis and processing were predominantly focused on hand-crafted features or shadow networks, due to limited training data we had. However, the status started to change as we have witnessed the significant growth of 3D data during the past few years (e.g., Warehouse 3D [1] and Yobi3D [2]), which offers rich opportunities for developing deep learning algorithms to significantly boost the performance of 3D data understanding.

Deep neural networks usually take vectorized data as input. This means how to encode input objects in vectorized forms is crucial to the resulting performance. While this problem is trivial for other modalities, e.g., audio signals, images and videos are already in vectorized forms, it becomes far more complicated for 3D objects, which are intrinsically 2D objects embedded in 3D ambient spaces. Existing deep learning algorithms fall into two categories, 3D-based and image-based. 3D-based methods typically encode a given 3D object using an occupancy grid. Yet due to memory and computational restrictions, the resolution of these occupancy grids remains low (e.g., 40x40x40 at best among most existing methods). Such a low resolution prohibits the usage of local geometric details for geometric understanding. In contrast to 3D-based approaches, image-based methods analyze and process 3D models via their 2D projections. Image-based techniques have the advantage that one can utilize significantly higher resolution for analyzing projected images. Moreover, it is possible to utilize large-scale training data (e.g., ImageNet) for pre-training. It turns out that with similar network complexity, image-based techniques appear to be superior to 3D-based techniques. Yet, there are significant restrictions of existing image-based techniques. For example, we need to determine the viewpoints for projection. Moreover, the projected images process discontinuities across image boundaries. Finally, existing techniques do not capture dependencies across different views, e.g., the correlation between the front and the back of an object, for geometric understanding.

[1] https://3dwarehouse.sketchup.com/?hl=en
[2] https://www.yobi3d.com/

In this paper, we introduce a novel projection method, which possesses the advantages of existing image-based techniques yet addresses the issues described above. The basic idea is to project 3D objects on a viewing sphere. On one hand, spherical domains are locally 2D, so that we can develop convolutional kernels at high resolution and utilize large-scale image data for pre-training. On the other hand, spherical domains are continuous and global, allowing us to capture patterns from complete 3D objects that are usually not present in standard image-based projections. Such characteristics make spherical projection advantageous compared with standard image-based projections. To fully utilize large-scale image training data, we present two spherical projection methods, one captures the depth variance in shapes from different view points, and the other captures the contour information of shapes from different view-points. These two projections utilize the textural and object boundary information that is captured by pre-trained neural network models from ImageNet.

We introduce two principled ways to utilize these spherical projections for the task of 3D object classification. The guiding principle is to perform convolutions on cylindrical patches, which admit standard 2D neural network operators and thus allow us to use pre-trained neural networks. We show how to sample cylindrical patches that minimize the number of cylindrical patches, and yet are sufficient to capture rich cross-view dependencies.

We have evaluated the proposed classification network on ModelNet40 [27] and ShapeNetCore [4]. Experimental results show that the proposed approach leads to results that are superior to or competing against state-of-the-art methods.

2. Related Works

3D object classification has been studied extensively in the literature. While early works focus on leveraging handcrafted features for classification [14, 1, 12, 8, 15, 9], recent methods seek to leverage the power of deep neural networks [27, 25]. For simplicity, we only provide a summary of methods that use deep neural networks, as they are most relevant to the context of this paper. Existing deep learning methods for 3D object classification fall into two categories: 3D-based and image-based.

3D-based methods classify 3D shapes based on 3D geometric representations such as voxel occupancy grids or point clouds. In [27], Wu et al. propose to represent a 3D shape as a probability distribution of binary variables on a 3D voxel occupancy grid and apply Deep Belief Network for classification. Recent methods utilize the same data representation but apply 3D convolutional neural networks for classification [13, 5, 2, 17, 3]. They differ from each other in terms of specifications of the training data (e.g., with front orientation or without front orientation) as well as details of network training. ORION [2] adds an orientation estimation module to the original VoxNet [13] as another task and trains both tasks simultaneously, which boosts the performance of VoxNet. Volumetric CNN [17] proposes two approaches to improve the performance of volumetric convolutional neural networks. The first one adds a sub-volume supervision task, which simultaneously trains networks that understand object parts as well as a network that understands the whole object. The second approach exploits an anisotropic probing kernel, which serves as a projection operator from 3D objects to 2D images. The results of the 2D projections can then be classified using 2D CNNs. The difference between our method and that of Volumetric CNN lies in the representation used for integrating 2D and 3D training data. Voxception-ResNet [3] designs a volumetric residual network architecture. To maximize the performance, it augments the training data with multiple rotations of the input and aggregates predictions from multiple residual networks. In addition to the representations of 3D convolution neural networks, people have proposed Beam Search [28] to optimize the model structure and hyper-parameters of 3D convolutional networks. The basic idea is to define primitive actions to alter the model structures and hyper-parameters locally, so as to find the best model structure and parameter setting for 3D objects represented by 3D voxel grids.

Despite the significant advances in 3D convolution, existing techniques possess a major limitation — the resolution of a 3D convolutional neural network is usually very coarse. Although this issue has been recently alleviated by Octree-based representations [19, 20, 26], the cost of 3D volumetric neural networks is still significant higher than 2D neural networks of the same resolution.

Besides voxel-based representations, people have looked at other geometric representations such as point-based representations. In [16], the authors propose a novel neural network architecture that operates directly on point clouds. This method leads to significant improvement in terms of running time. The major challenge of designing neural networks for point cloud data is to ensure permutation invariance of the input points. In an independent work, SetLayer [18] concentrates on the permutation equivalent property of point cloud and introduces a set-equivariant layer for this purpose.

Image-based techniques. The key advantage of 3D-based techniques is that the underlying 3D object can be exactly characterized by the corresponding 3D representation. On the other hand, 3D training data remains limited compared to the amount of 2D training data we have access to. In addition, such 3D representations are unable to utilize the large amount of labeled images, which can be considered as projections of the underlying 3D objects. This is the motivation of image-based 3D object classification techniques, which apply 2D convolutional neural networks to classify

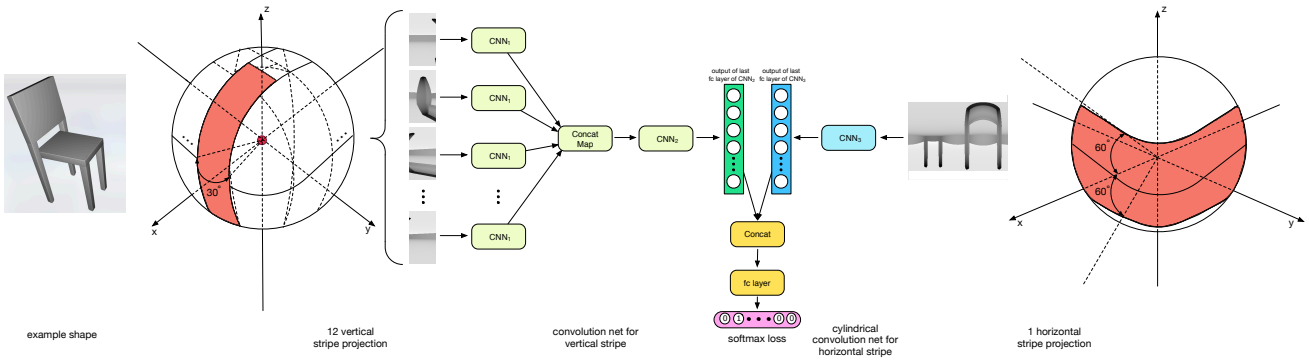


Figure 1: Illustration of the depth-based projection network. The network takes a spherical projection of the an input object as input. It applies convolution operations on cylindrical strips. The output of these sub-networks are passed through a fully connected module, which captures data dependencies across different strips.

rendered images. In [25], Su et al. propose to render 12-views for 3D shapes and classify the rendered images. The image classification network is initialized using VGG [23], pretrained on ImageNet data and then fine-tuned on the ModelNet40 dataset. [10] provides a different way to fine-tune the network with rendered images, each of which is given a weight to measure its importance to the final prediction. [24] proposes a way to convert 3D objects to geometry images and implicitly learn the geometry and topology of 3D objects. Despite the fact that image-based techniques can utilize pre-trained image classification networks, image projections present significant information loss, and it is not easy to capture complete relative dependencies, e.g., those that can not be projected to the same view. Perhaps the most relevant to our method is classifying panoramas [22, 21], which projects a 3D object onto a cylindrical domain. Although cylindrical domain is certainly more flexible than image-domains, it still does not cover the entire object. Our experimental results reveal that a single cylindrical projection is insufficient for obtaining state-of-the-art object classification performance.

Hybrid methods. Several works seek to combine 3D-based techniques and image-based techniques. In particular, FusionNet [6] utilizes both 3D voxel data and 2D projection data by training two 3D CNNs: general 3D CNN and kernel-based 3D CNN with varying size. FusionNet also trains an image-based multi-view CNN network mentioned above. FusionNet then fuses features from these three networks so as to exploit advantages of different features.

We observe that methods based on 2D projections tend to perform better than those based on volumetric representations since they can exploit pre-trained models to address the issue of insufficient training data. Yet, image-based techniques require a large number of views. There are significant overlaps across views, exhibiting information redundancy. Thus, we propose a novel spherical projection approach, which uses a single sphere to aggregate information from different viewing directions. We also explore data dependencies across different views, which are beneficial for object classification.

3. Spherical Projection

In this section, we describe the two proposed spherical projections, i.e., depth-based projection and image-based projection. The input to these two projections is a 3D model with prescribed upright orientation. However, we do not assume the front orientation is given. Such a setup is applicable to almost all internet 3D model repositories (e.g., Warehouse3D and Yobi3D). Both spherical projections utilize a sphere centered around the barycenter of each object. The radius of this sphere is chosen as three times the diagonal of the object bounding box. Note that the radius of the sphere does not affect the depth-based projection and has minor effects on the image-based projection.

Depth-based projection. The depth-based projection is generated by shooting a ray from each point on the sphere to the center. Each point is recorded as the distance to the first hitting point. Otherwise, the distance is set to be zero. We compute depth values for vertices of a semi-regular quad-mesh whose axis aligns with the longitude and latitude, i.e.,

$$\begin{aligned} &(\cos(\theta_i)\cos(\phi_j), \cos(\theta_i)\sin(\phi_j), \sin(\theta_i)) \\ &\theta_i = \frac{180^\circ \cdot i}{m}, \phi_j = \frac{360^\circ \cdot j}{n}, \quad \begin{matrix} 0 \le i \le m-1 \\ 0 \le j \le n-1 \end{matrix} \end{aligned} \tag{1}$$

Then the depth value of other points on the sphere are generated by linear interpolation. Specifically, denote d_{ij} as the depth value that corresponds to (θ_i, ϕ_j). Then given a point with spherical coordinate (θ, ϕ), where $\theta_i \le \theta \le \theta_{i+1}, \phi_j \le$

$\phi \leq \phi_{j+1}$, its depth value is given by

$$\begin{aligned} d =&(1-t_{ij})\big((1-s_{ij})d_{ij} + s_{ij}d_{i,j+1}\big) \\ &+ t_{ij}\big((1-s_{ij})d_{i+1,j} + s_{ij}d_{i+1,j+1}\big). \end{aligned} \tag{2}$$

This allows us to generate the depth value for every single point on the sphere. In our implementation, we further use an Octree to accelerate the ray-mesh intersection. For all of experiments, we use $n = 180, m = 90$, i.e., one pixel per 2 degrees along both the latitude and longitude. We proceed to generate cylindrical strips from the depth projection described above. We first use the strip covering the following area:

$$\begin{aligned} &(\cos(\theta_{i_h})\cos(\phi_{j_h}), \cos(\theta_{i_h})\sin(\phi_{j_h}), \sin(\theta_{i_h})) \\ &\theta_{i_h} = \frac{120^\circ \cdot i_h}{m_h} + 30^\circ, \phi_{j_h} = \frac{360^\circ \cdot j_h}{n_h}, \quad \begin{matrix} 0 \leq i_h \leq m_h - 1, \\ 0 \leq j_h \leq n_h - 1. \end{matrix} \end{aligned} \tag{3}$$

Since regions of high latitude suffer from severe distortion, we eliminate them by setting θ_{i_h} from 30° to 150°. In the following, we will call this strip the latitude strip, which is fitted into the convolution layers (See Figure 1).

To utilize information form high latitude regions for classification, we also utilize a circle of vertical strips parallel to a longitude (See Figure 1). There are 12 strips in total, and the angle between adjacent strips is 30°. The pixel coordinates on each strip indexed by k_v is given by

$$\begin{aligned} &(\cos(\delta_{k_v})\cos(\theta_{i_v})\sin(\phi_{j_v}) - \sin(\delta_{k_v})\sin(\theta_{i_v}), \\ &\cos(\delta_{k_v})\sin(\theta_{i_v}) + \sin(\delta_{k_v})\cos(\theta_{i_v})\sin(\phi_{j_v}), \\ &\cos(\theta_{i_v})\cos(\phi_{j_v})) \\ &\theta_{i_v} = \frac{360^\circ \cdot i_v}{l_v m_v} + 90^\circ - \frac{180^\circ}{l_v}, \phi_{j_v} = \frac{180^\circ \cdot j_v}{n_v}, \delta_{k_v} = \frac{360^\circ \cdot j_v}{l_v} \\ &0 \leq i_v \leq m_v - 1, 0 \leq j_v \leq n_v - 1, 0 \leq k_v \leq l_v - 1 \end{aligned} \tag{4}$$

where l_v is the number of strips. For all experiments shown in this paper, we use $n_h = 360, m_h = 240$ and $n_v = 180, m_v = 60, l_v = 12$. The total number of pixels is comparable with that used in the MVCNN [25].

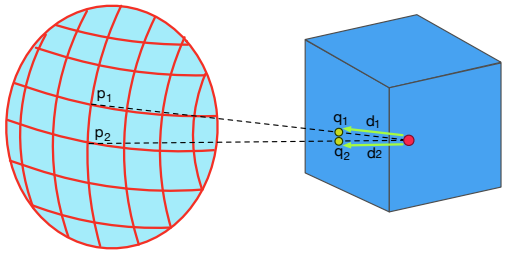


Figure 2: Illustration of the depth-based projection method. We compute depth values on a rectangular grid, which is then used for interpolation.

As illustrated in Figure 2, depth-based projection effectively captures geometric variations. Moreover, for a wide range of objects (i.e., the ray defined by every shape point and the sphere center reaches the sphere without occlusion), the original shapes can be directly reconstructed from the depth-based projection. Such objects include convex objects and many other box-like objects. In other words, depth-based projection is quite informative. On the downside, at the global scale, the pattern reveals in the depth-based projection seems to deviate from natural images. Moreover, the contours of objects, which provide important cues for 3D object classification, are not present in the depth-based projection. This motivates us to consider image-based projection.

Image-based projection. As shown in Figure 3, image-based projection shoots a 3x12 grid of images of the input object from 36 view points in total. The locations of the cameras are given by setting $m = 12, n = 3$ in (1), i.e., $\phi_j = 0^\circ, 30^\circ, \cdots, 330^\circ, \theta_i = -60^\circ, 0^\circ, 60^\circ$. At each camera location, the up-right orientation of the camera always points to the north-pole. The viewing angle of each image is 45° so that the projected images barely overlap. The resolution of each image is 224x224. In our experiments, we have varied the value of m and found that $m = 12$ provides a good trade-off between minimizing the number of views and ensuring that the resulting projections are approximately invariant to rotating the input object.

Note that our image-based projection does not generate a per-pixel value for each point on the sphere. Instead, we use the sphere to guide how these images are projected, enabling us to capture dependencies across different views.

4. Classification Network and Training

The major motivation of the proposed network design is two-fold: 1) leveraging pre-trained convolution kernels from ImageNet, and 2) capturing dependencies that cannot be projected in the same view (e.g., front and back of an object). To this end, we propose two steps for designing the classification network.

Network design. Figure 1 illustrates the network-design for depth-based projection. The same as AlexNet, this network has a convolution module and a fully connected module. The convolution module utilizes the same set of convolution kernels as AlexNet. This allows us to use the pre-trained kernels from AlexNet. As shown in Figure 1, if the strip is parallel to the latitude, the convolutions are applied in a periodical manner, so as to utilize the continuity of the data. Specifically, CNN_3 is a periodical convolution network which captures dependencies across different views in both convolutional and fully-connected layers. In contrast, common convolutions are applied to strips that are parallel to the longitude independently (See CNN_1). The fully connected layers, which are shown in CNN_2, capture the data

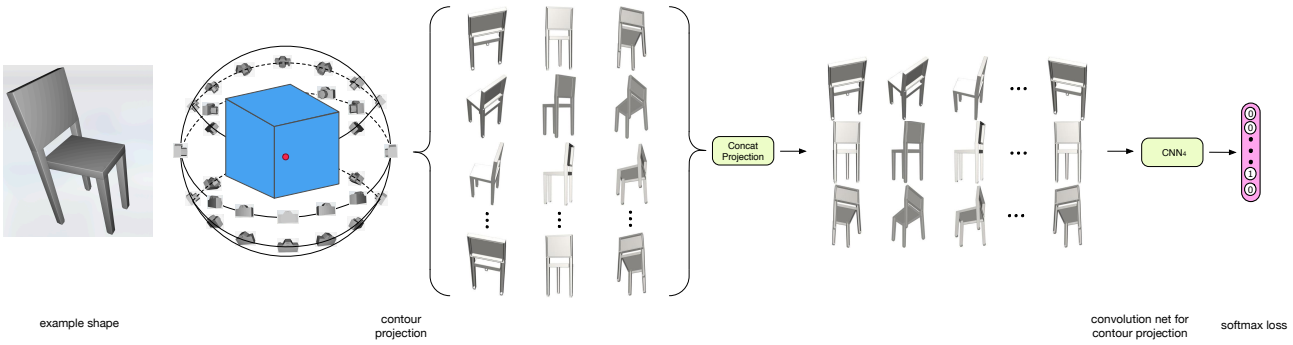


Figure 3: Illustration of the contour-based projection. We utilize 36 rendered images arranged in a grid. The convolution operators are applied on discrete cylindrical strips generated from this 2D array.

dependencies across different views. To preserve the spatial relation while maintain rotation invariance, we introduce 12 fully connected layer-connections $(W_k, \mathbf{b}_k), 0 \leq k \leq 11$ between CNN_1 and CNN_2. Let $\mathbf{f}_i^l, 0 \leq i \leq 11$ be the feature vectors at layer l. We define the feature vectors at layer $l+1$ as

$$\mathbf{f}_i^{(l+1)} = \sigma(\sum_{j=1}^{12} W_{(i-j) \bmod 12} \mathbf{f}_j + \mathbf{b}_{(i-j) \bmod 12}).$$

Note that the initial network weights $(W_0, \mathbf{b}_0)$ are set to be the AlexNet weights. The other weights are initialized as zero, i.e., $W_i = 0, \mathbf{i} = 0, 1 \leq i \leq 11$. In other words, the initial weights apply fully connected operations on the feature vector attached to each image in isolation, while the cross links force the network to learn dependencies across different views.

Figure 3 shows the network design for contour-based projections. 36 Cameras are uniformly distributed along three latitudes (60°, 90°, 120°) of the sphere. We concatenate all the rendered images in their spatial order on the sphere. We then feed the entire image to CNN_4, i.e., the convolutional neural network for the contour-based projection.

As described in the previous Section, the resolutions of the depth-based projections are 240x360 and 360x60x12 for strips along the longitude and the latitude, respectively. In addition, the resolution of the contour-based projection is 224x224x36. Note that although we utilize more pixels, the number of parameters in the network remains relatively small, as we share network parameters across different strips.

Training. We train the entire depth network at three stages. We first train the convolutional layers and the direct connection layers, i.e., $(W_0, \mathbf{b}_0)$. If pre-training is allowed, the training at this stage starts from the pre-trained weights of AlexNet. We then train the convolutional network for the latitude strip. After this step, we have two pre-trained models for longitude and latitude strips. Finally, we train the entire network together with weights copied from these two pre-trained models except the final classifying layer. Note that for the contour-based network, we directly train from scratch if AlexNet parameters are not provided. If AlexNet parameters are provided, we train with all the parameters except the last classifying layer.

We use the Caffe Framework for all of our experiments. For layers which are trained from scratch, we set its learning rate to be 10 times that of the other layers. We used mini-batch stochastic gradient descent (SGD) with 0.9 momentum and the learning rate annealing strategy implemented in Caffe. The learning rate is is cross-validated by grid search started from 10^{-5} and ended at 10^{-2}, where the multiplicative step-size is $10^{\frac{1}{2}}$. We fix the mini-batch size as 32 and set the weight decay as 0.0005.

5. Experimental Evaluation

5.1. Experimental Setup

Datasets and evaluation propocol. We evaluate the proposed approaches on two Benchmark datasets ModelNet40 and ShapeNetCore. They both collect models from Warehouse3D but with different model classes.

ModelNet40 [27] contains 12311 shapes across 40 categories ranging from airplane to xbox. We use the default training-testing split (c.f. [13, 2, 17]) that leads to 9843 models in the training set and 2468 models in the testing set.

ShapeNetCore [4] contains 51300 shapes across 55 categories. We use the default training set (36148 models) for training and default validation set (5615 models) for testing. Note that the size of ShapeNetCore is bigger than that of ModelNet40. Distributions of categories are also different, e.g., ShapeNetCore contains less furniture categories.

Baseline methods. Since our method does not utilize the front orientation, for baseline comparison we only consider algorithms that do not utilize such information as well. In addition, we also report performance of state-of-the-art methods on ModelNet40.

The baseline algorithms we choose include MVCNN [25],

Table 1: Accuracy on ModelNet40 and ShapeNetCore of our approaches and the various baseline methods. We report the performance on these two entire datasets and subsets as well as two curated subsets.

Method	ModelNet40	ShapeNetCore	ModelNet40-SubI	ShapeNetCore-SubI
3D Shapenets [27]	85.9	na	83.33	na
Voxnet [13]	87.8	na	85.99	na
FusionNet [6]	90.80	na	89.54	na
Volumetric CNN [17]	89.9	na	88.65	na
MVCNN [25]	92.31	88.93	91.22	88.64
MVCNN-MultiRes [17]	93.8	90.01	92.60	90.00
OctNet [20]	87.83	88.03	86.45	87.85
depth-base pattern	91.36	89.45	90.25	89.13
contour-based pattern	93.31	90.49	92.20	90.80
overall pattern	94.24	91.00	93.09	91.22

MVCNN-MultiRes [17], 3D ShapeNets [27], Voxnet [13], FusionNet [6], Volumetric CNN [17] and OctNet [20]. In the following, we briefly summarize the characteristics of these methods. MVCNN classifies a given model by fusing the feature layers of rendered images with a max-pooling layer. MVCNN-MultiRes improves MCVNN by exploiting rendered images from multiple resolutions. 3D ShapeNets is the first deep learning method on 3D shape data which is built on a Deep Belief Network. Voxnet leverages a 3D convolution neural network for shape classification. FusionNet fuses features extracted from 3D voxel data and 2D projection data by different networks. Finally, Volumetric CNN modifies Voxnet by adding subvolume supervision task and anisotropic probing kernel convolution. OctNet transforms 3D objects into Octree-based representations and design a special network to classify these representations. All of these methods use the upright orientation but do not use the front orientation.

5.2. Classification Results

Table 1 collects the overall classification accuracy of different methods on ModelNet40 and ShapeNetCore. As we can see, the proposed depth-based projection method is superior to most existing 3D-based methods. This demonstrates the power of incorporating massive image training datasets. Compared to most other image-based techniques, our contour-driven projection method exhibits the top performance, showing the advantage of generating projections on the spherical domain. Although MVCNN-MultiRes outperforms contour-based projection, it needs to render images of different resolutions, which is much slower than our contour projection. When combining depth-based projection and contour-based projection, our overall method performs better than MVCNN-MultiRes and achieves the highest accuracy, which also demonstrates that our two projections are complementary.

When comparing the performance of various algorithms on ModelNet40 and ShapeNetCore, we find that the performance on ShapeNetCore is lower, which is expected since repositories in ShapeNetCore exhibit bigger variance. Moreover, as ShapeNetCore is bigger than ModelNet40, the gap between depth-based projection and view-based project is bigger, since the effects of pre-training may be reduced when the size of the 3D data increases. In the following, we provide more detailed analysis of the results.

5.3. Analysis of Results

The effects of pre-training. As illustrated in Table 2 and Table 3, all 2D-based techniques benefit from ImageNet pre-training which justifies the fact that the size of both ShapeNetCore and ModelNet40 are relative small to train high-quality classification networks, and images from ImageNet contain rich information that can be used to differentiate rendered images. When comparing ShapeNetCore with ModelNet40, the effects of pre-training on ShapeNetCore is more salient than that on ModelNet40. An explanation is that ShapeNetCore exhibits higher diversity, so that ImageNet features help more. Another factor is that the distribution of ShapeNetCore categories are closer to corresponding categories in ImageNet than ModelNet40 categories.

Quite surprisingly, the improvement of pre-training on depth-based projection is as strong as that on contour-driven projection. This means that pre-trained ImageNet models contain rich interior edge information as well (e.g., changes of texture information in the presence of depth continuities), which is beneficial for classifying depth-based projections.
Depth-based versus contour-based. The overall performance of depth-based projection is slightly below that of contour-driven projection. This is expected because object contours provide strong cues for classification.

To further compare the effectiveness of different methods on a particular type of shapes, we selected the classes belonging to furniture from both datasets as two curated subsets, which are bathtub, bed, bookshelf, chair, curtain, desk, door, dresser, lamp, mantel, night_stand, range_hood, sink, sofa, stool, table, toilet, tv_stand, wardrobe in ModelNet40 and bathtub, bed, bookshelf, cabinet, chair, clock, dishwasher, lamp, loudspeaker, sofa, table, washer in ShapeNetCore re-

Table 2: Accuracy Before and After Pre-training on ModelNet40

Method	Before Pre-training		After Pre-training	
	Accuracy (class)	Accuracy (instance)	Accuracy (class)	Accuracy (instance)
MVCNN	82.15	87.15	90.35	92.31
MVCNN-MultiRes	88.12	91.20	91.40	93.80
depth-base pattern	80.44	86.09	87.32	91.36
contour-based pattern	88.33	91.48	91.16	93.31
overall pattern	88.53	91.77	91.56	94.24

Table 3: Accuracy Before and After Pre-training on ShapeNetCore

Method	Before Pre-training		After Pre-training	
	Accuracy (class)	Accuracy (instance)	Accuracy (class)	Accuracy (instance)
MVCNN	67.84	84.55	78.79	88.93
MVCNN-MultiRes	75.34	88.4	79.01	90.01
depth-base pattern	70.55	85.15	78.84	89.45
contour-based pattern	74.52	88.54	79.38	90.49
overall pattern	75.60	88.87	80.38	91.00

Table 4: Accuracy of Each Class For Different Projection on ModelNet40

Class Name	Depth-Based	Contour-Based	MVCNN	Class Name	Depth-Based	Contour-Based	MVCNN
bowl	100.00	95.00	85.00	stool	75.00	75.00	75.00
bookshelf	99.00	99.00	94.00	tent	95.00	95.00	95.00
cone	100.00	100.00	95.00	toilet	100.00	100.00	100.00
table	89.00	84.00	84.00	xbox	80.00	80.00	80.00
vase	82.00	85.00	77.00	car	99.00	100.00	100.00
tv_stand	85.00	90.00	81.00	guitar	98.00	100.00	99.00
dresser	89.53	89.53	86.05	monitor	97.00	99.00	99.00
bottle	98.00	96.00	96.00	plant	85.71	89.80	87.76
sofa	98.00	99.00	97.00	range_hood	93.00	97.00	96.00
airplane	100.00	100.00	100.00	night_stand	75.58	86.05	80.23
bathtub	94.00	96.00	94.00	sink	85.00	85.00	90.00
bed	100.00	100.00	100.00	piano	91.00	96.00	97.00
bench	80.00	80.00	80.00	mantel	93.00	97.00	100.00
chair	98.00	99.00	98.00	curtain	85.00	95.00	95.00
desk	86.05	87.21	86.05	lamp	75.00	80.00	85.00
door	100.00	100.00	100.00	cup	55.00	80.00	70.00
glass_box	97.00	97.00	97.00	flower_pot	15.00	15.00	30.00
keyboard	100.00	100.00	100.00	wardrobe	65.00	90.00	90.00
laptop	100.00	100.00	100.00	radio	65.00	95.00	95.00
person	100.00	95.00	100.00	stairs	70.00	100.00	100.00

spectively. We tested our model on these subsets, and results are included in Table 4.

It is clear that the winning categories of each method are drastically different. As indicated in Table 4, depth-based projection is advantageous on categories such as bowl, table, and bottle, which possess strong interior depth patterns. In contrast, contour-based projection is superior to depth-based projection on categories such as plant, guitar, and sofa, which have unique contour features.

Comparison to MultiView-CNN. The proposed contour-based projection method is superior to MultiView-CNN on both ModelNet40 and ShapeNetCore. The main reason is due to the fact our network captures dependencies earlier in the convolution, while MultiView-CNN only max-pools features extracted from the convolution layers. This performance gap indicates that dependencies across different views and at different scales are important cues for shape classification.

Comparison to voxel-based techniques. Our approach is also superior to most voxel-based classification methods, indicating the importance of leveraging image training data. The only exception is the recent work of Voxel-ResNet [28]. However, that work assumes that the front-orientation of each shape is given. In addition, its performance highly relies on training an ensemble network. In contrast, the accuracy of each individual network of [28] is upper bounded by 90.1%.

Comparison to panorama-based techniques. A building block of our technique is based on classifying cylindrical strips of spherical projections. This is relevant to some recent works of classifying panoramas of 3D objects [22, 21]. However, the major difference in our approach is that we use multiple strips to capture the correlations of the spherical projection from multiple strips. In addition, the network design utilizes a pre-trained model from ImageNet. As indicated by our experiments, on ModelNet40 our approach leads to 4.5% and 3.4% improvements in terms of accuracy

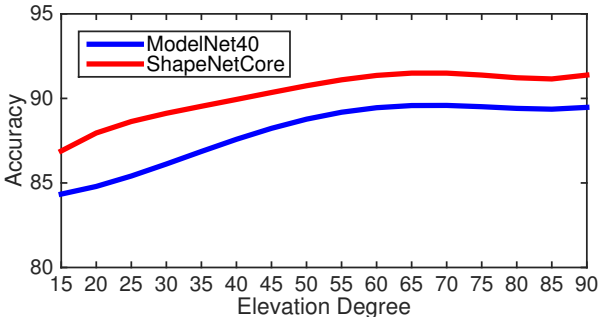
from using a single strip and that of [21], respectively.

Table 5: Accuracy w.r.t Number of Views for Depth and Contour Pattern on ModelNet40 and ShapeNetCore

Pattern	Number of Views	ModelNet40	ShapeNetCore
depth-based	6	90.87	88.95
	24	91.02	89.23
	12	91.36	89.45
contour-based	6	92.26	89.23
	24	93.12	90.36
	12	93.31	90.49

Varying the resolutions of the projections. We have also tested the performance of our network by varying resolutions of the projections. For depth-based projection, we increase the resolution of the grid from 30° to 15° and 60°, we found that the classification accuracy drops by 0.2% and 0.5%, respectively. We thus used 30° for efficiency concerns. For contour-based projection, we have changed resolution of the grid pattern to 3×24 and 3×6, we found that the improvements in classification accuracy improves by less than 0.2% from 3×12 grids to 3×24 grids. On the other hand, the improvement from using 3×6 grids to 3×12 grids is about 1.0% on average, which is expected since using a 3×6 grid is insufficient for handling rotation invariance.

Figure 4: Accuracy w.r.t Elevation degree of the strip parallel to the latitude



Varying the elevation degree horizontal strip. We also tried varying the elevation degree of the horizontal strips, which is 60° in Figure 1. Experimental results in Figure 4 show that the performance is not that sensitive when the elevation degree varies. We can see that when the elevation degree increases larger than 60°, the performance does not apparently increase with the increasing of the elevation degree, since high altitude areas have severe distortion in the horizontal strip, where pre-trained models become ineffective. Thus, we choose elevation degree as 60°.

Timing. The rendering, inference, and training time are listed in Table 6, which are performed on a machine with 16 Intel Xeon E5-2637 v4 @ 3.50GHz CPUs and 1 Titan X(Pascal) GPU. As indicated in Table 6, classifying a single 3D object takes around 2 seconds. The dominant computational cost is on generating the projections. Note that both depth-based projections and contour-based projections can be accelerated using GPU. We believe the computational cost can be significantly improved by exploring such options.

Table 6: Running Time of All the methods on ModelNet40 and ShapeNetCore

Method	ModelNet40			ShapeNetCore		
	Rendering	Inference	Training	Rendering	Inference	Training
Depth	0.92s	0.043s	252m	1.06s	0.043	272m
Contour	1.17s	0.418s	371m	1.28s	0.418s	442m

6. Conclusions, Discussion and Future Work

In this paper, we have introduced a spherical representation and developed deep neural networks to classify 3D objects. Our approach explores two ways to project 3D shapes onto a spherical domain. The first one leverages depth variation, while the other one leverages contour information. Such a representation shows advantages of 1) allowing high resolution grid representations to capture geometric details, 2) incorporating large-scale labeled images for training, and 3) capturing data dependencies across the entire object. We also described principled ways to define convolution operations on spherical domains such that the output of the neural networks is not sensitive to the front orientation of each object. Experimental results show that the proposed methods are competitive against state-of-the-art methods on both ModelNet40 and ShapeNetCore.

There are ample opportunities for future research. The methods presented in this paper still use rectangular convolutional kernels mainly due to the fact that we want to use pre-trained convolutional kernels. However, technically it will be interesting to see if one can define convolutional kernels directly on spherical domains. One potential solution is to use spherical harmonics [11]. In another direction, it remains interesting to consider other types of spherical projections, e.g., spherical parameterizations of geometric objects [7], which is free of occlusions. We did not use such parameterizations mainly due to that models in ModelNet40 and ShapeNetCore consist a lot of disconnected components. Finally, we only consider the task of classification, it will be interesting to consider other tasks such as shape segmentation and shape synthesis. For both tasks, the standard image-based techniques require stitching predictions from different views of the objects. In contrast, the spherical projection is complete and may not suffer from this issue.

Acknowledgement. We would like to acknowledge support of the NSF Award IIP #1632154. We also gratefully acknowledge the support of NVIDIA Corporation with the donation of the Titan Xp GPU used for this research. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors.

References

[1] The princeton shape benchmark. In *Proceedings of the Shape Modeling International 2004*, SMI '04, pages 167–178, Washington, DC, USA, 2004. IEEE Computer Society.

[2] N. S. Alvar, M. Zolfaghari, and T. Brox. Orientation-boosted voxel nets for 3d object recognition. *CoRR*, abs/1604.03351, 2016.

[3] A. Brock, T. Lim, J. Ritchie, and N. Weston. Generative and discriminative voxel modeling with convolutional neural networks. *arXiv preprint arXiv:1608.04236*, 2016.

[4] A. X. Chang, T. A. Funkhouser, L. J. Guibas, P. Hanrahan, Q.-X. 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. *CoRR*, abs/1512.03012, 2015.

[5] A. Garcia-Garcia, F. Gomez-Donoso, J. G. Rodríguez, S. Orts-Escolano, M. Cazorla, and J. A. López. Pointnet: A 3d convolutional neural network for real-time object class recognition. In *2016 International Joint Conference on Neural Networks, IJCNN 2016, Vancouver, BC, Canada, July 24-29, 2016*, pages 1578–1584, 2016.

[6] V. Hegde and R. Zadeh. Fusionnet: 3d object classification using multiple data representations. *CoRR*, abs/1607.05695, 2016.

[7] K. Hormann, K. Polthier, and A. Sheffer. Mesh parameterization: Theory and practice. In *ACM SIGGRAPH ASIA 2008 Courses*, SIGGRAPH Asia '08, pages 12:1–12:87, New York, NY, USA, 2008. ACM.

[8] Q.-X. Huang, H. Su, and L. Guibas. Fine-grained semi-supervised labeling of large shape collections. *ACM Trans. Graph.*, 32(6):190:1–190:10, Nov. 2013.

[9] N. Iyer, S. Jayanti, K. Lou, Y. Kalyanaraman, and K. Ramani. Three-dimensional shape searching: State-of-the-art review and future trends. *Comput. Aided Des.*, 37(5):509–530, Apr. 2005.

[10] E. Johns, S. Leutenegger, and A. J. Davison. Pairwise decomposition of image sequences for active multi-view recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 3813–3822, 2016.

[11] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *Symposium on Geometry Processing*, SGP '03, pages 156–164, 2003.

[12] J. Knopp, M. Prasad, G. Willems, R. Timofte, and L. Van Gool. Hough transform and 3d surf for robust three dimensional classification. In *Proceedings of the 11th European Conference on Computer Vision: Part VI*, ECCV'10, pages 589–602, 2010.

[13] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *IROS*, pages 922–928, 2015.

[14] R. Osada, T. Funkhouser, B. Chazelle, and D. Dobkin. Shape distributions. *ACM Trans. Graph.*, 21(4):807–832, Oct. 2002.

[15] J. Pu and K. Ramani. On visual similarity based 2d drawing retrieval. *Comput. Aided Des.*, 38(3):249–259, Mar. 2006.

[16] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, abs/1612.00593, 2016.

[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 *Computer Vision and Pattern Recognition*, pages 5648–5656, 2016.

[18] S. Ravanbakhsh, J. G. Schneider, and B. Póczos. Deep learning with sets and point clouds. *CoRR*, abs/1611.04500, 2016.

[19] G. Riegler, A. O. Ulusoy, H. Bischof, and A. Geiger. Octnetfusion: Learning depth fusion from data. *CoRR*, abs/1704.01047, 2017.

[20] G. Riegler, A. O. Ulusoy, and A. Geiger. Octnet: Learning deep 3d representations at high resolutions. *CoRR*, abs/1611.05009, 2016.

[21] K. Sfikas, T. Theoharis, and I. Pratikakis. Exploiting the PANORAMA Representation for Convolutional Neural Network Classification and Retrieval. In *Eurographics Workshop on 3D Object Retrieval*, 2017.

[22] B. Shi, S. Bai, Z. Zhou, and X. Bai. Deeppano: Deep panoramic representation for 3-d shape recognition. *IEEE Signal Process. Lett.*, 22(12):2339–2343, 2015.

[23] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[24] A. Sinha, J. Bai, and K. Ramani. Deep learning 3d shape surfaces using geometry images. In *European Conference on Computer Vision*, pages 223–240, 2016.

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

[26] M. Tatarchenko, A. Dosovitskiy, and T. Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. *CoRR*, abs/1703.09438, 2017.

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

[28] X. Xu and S. Todorovic. Beam search for learning a deep convolutional neural network of 3d shapes. *CoRR*, abs/1612.04774, 2016.