3D object classification and retrieval with Spherical CNNs

Carlos Esteves[1] Christine Allen-Blanchette[1] Ameesh Makadia[2] Kostas Daniilidis[1]
[1] GRASP Laboratory, University of Pennsylvania [2] Google

Abstract

3D object classification and retrieval presents many challenges that are not present in the traditional (planar) image setting. First, there is the question of shape representation. Face-vertex meshes, for instance, are widely used in computer graphics, but their irregular structure complicate their use as inputs to learning models. Previous works have converted meshes to more structured representations, such a collections of rendered views or volumetric grids, in order to feed them to 2D or 3D CNNs. These representations, however, are redundant and wasteful, requiring large amounts of storage, pre-processing time, and large networks with millions of parameters to handle them. Another challenge is how to treat object orientations. Orientation-invariance is a desired property for any classification engine, yet most current models do not address this explicitly, rather requiring increased model and sample complexity to handle arbitrary input orientations. We present a model that aims to be efficient in both the number of learnable parameters and input size. We leverage the group convolution equivariance properties; more specifically, the spherical convolution, to build a network that learns feature maps equivariant to SO(3) actions by design. By mapping a 3D input to the surface of a sphere, we also end up with a small input size.

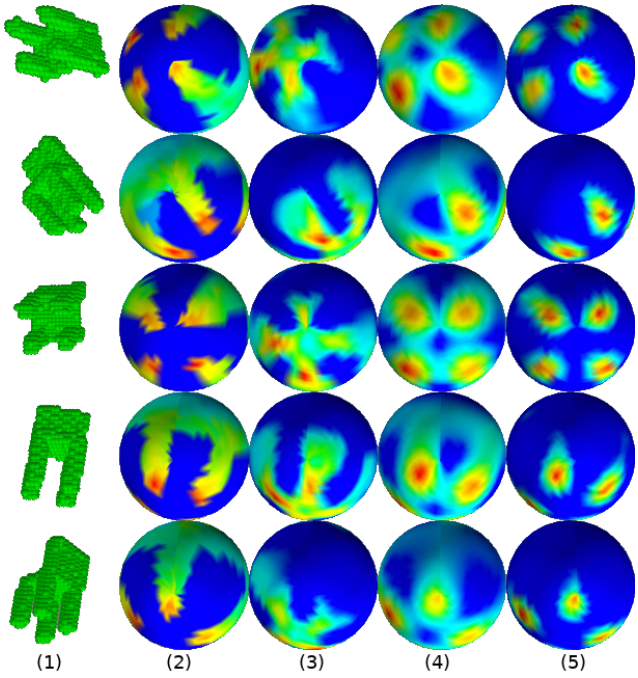

Figure 1. Columns, left to right: (1) input rotations, (2) initial spherical representation, (3, 4, 5) learned feature maps. Note how the features maps are equivariant to 3D rotations. Observe the activations on the chair legs direction for easier verification.

1. Introduction

With increased accessibility of 2.5D scanners (e.g. Microsoft Kinect, Intel RealSense) and 3D modeling software (e.g. Autodesk, SketchUp), the corpus of (freely) available 3D models is growing rapidly (e.g. 3D Warehouse); however, 3D datasets are still nowhere near as large or prolific as 2D image datasets, ModelNet [31] contains 12K 3D models while ImageNet [27] contains 14M 2D images. As a result, the development of repeatable and discriminative 3D representations significantly lags their 2D counterparts.

Spherical CNN, the subject of this work, is an effort toward the development of 3D model representations which are simultaneously discriminative and invariant in the presence of 3D object symmetry deformations. An object's symmetry deformations are deformations which do not change the innate character of the object (e.g. translations). Enumeration of all 3D object symmetries is nontrivial since many may be class dependent or otherwise difficult to characterize; however, there is a general consensus on the in variance of an object's class to variations in scale, translation and rotations.

Normalization of translation and scale can be achieved by setting the object's origin to its center of mass and constraining the extent of the object to a fixed constant. Normalization of rotation, however, has been found to be a sig-

http://github.com/daniilidis-group/spherical-cnn

nificant challenge. This issue of rotation normalization impacts the ability to generate discriminative rotationally invariant object descriptors which in turn contributes to variations in an object's representation in response to viewpoint variation and subsequent inaccuracies in classification and retrieval tasks.

This work proposes to resolve the issue of rotation normalization using Spherical CNN, a new approach to convolutional neural networks (CNNs) which learns a rotationally invariant spherical representation of a 3D object model. Spherical CNN takes as input a hand-crafted spherical representation of a 3D model which is inherently equivariant to rotational deformation. Application of cascaded spherical convolutions, nonlinearities and pooling maintains rotational equivariance as is evidenced by intermediate representations of Spherical CNN (see Figure 1). The output of Spherical CNN is a learned rotationally invariant representation (see Figure 5). A learned rotationally equivariant spherical representation can be retrieved from feature maps in the second to last layer (see Figure 2). The utility of this representation is demonstrated on a number of classification and retrieval (i.e. ModelNet10, ModelNet40 [31], ShapeNet Core55 [7]) datasets. Concretely, our contributions are

- a learned 3D rotationally equivariant representation of 3D objects and spherical functions
- an analysis of various spherical representations
- a study of various convolutional architectures for learning spherical representations

This document is organized as follows. Section 2 details representative work in 3D data representation including learned representations and describes their challenges and successes in classification tasks. Section 3 offers theoretical background for the design of Spherical CNN, most notably detailing group convolution. Specifics of the implementation including efficient computation of the spherical correlation are outlined in Section 4 and experimental results and insights are offered in Section 5.

2. Related Work

Representations of 3D data can be divided into those which are regular, meaning the structure of the representation is fixed, and those which aren't (i.e. irregular). The "raw" representation from a 2.5D sensor is an irregular point cloud since the number of points can vary both within and across object classes, and the representation is generally not equivariant to rotational deformation of the object. 3D models can also present as irregular since, similarly to point clouds, the number of vertices can vary within and across object classes due to changes in appearance and choice of tessellation.

There is a wide body of literature on constructing irregular representations of 3D surfaces by associating a descriptor to each vertex of a triangular mesh. The Heat Kernel Signature (HKS) [29] is constructed by computing the mesh Laplacian [3] over the triangular mesh and using a combination its eigenvalues and eigenvectors to generate a signature at each mesh vertex over a fixed time interval. The Intrinsic Shape Context descriptor (ISC) [18] is an extension of the 2D Shape Context [4] descriptor to triangular meshes. The log-polar descriptor is constructed by mapping the 1-neighborhood of each vertex to the plane where modified angular and radial statistics are computed. The descriptor is orientation normalized by taking the modulus of the descriptor's Fourier Transform.

While it is common for regular representations to be constructed over a triangular mesh, they contrast with irregular representations in that they are commensurable without knowledge of which inter-object descriptors are in the space of correspondences. A particularly popular regular representation is the voxel-grid which is formed by assigning a probabilistic value to each location of a 3D lattice in accordance with point-cloud or 3D mesh data. Shape Distributions [23], a global shape signature, is generated by computing the distribution of distances/angles between a fixed number of randomly selected surface points. Extended Gaussian Images [15] are obtained by mapping the inverse curvature of the surface to a location on the sphere determined by the surface normal. While the resolution of the information in this representation is partially determined by the resolution of the mesh, the dimension of the output is wholly determined by the choice of spherical tessellation.

A particularly nice property of the spherical representation is that pose registration can be achieved efficiently. Whereas naïvely computing the correlation of spherical functions requires $O(NL^5)$, employing harmonic analysis reduces the complexity to $O(L^3 \log^2 L)$ [19]. All representations described thus far are hand-crafted. Spherical CNN is set apart from these approaches in that the representations are learned from data.

Convolutional Neural Networks (CNNs) have demonstrated incredibly strong performance on 2D inputs over a myriad of tasks. Given this track record it is desirable to amend this powerful technology to accommodate 3D data. For classification, the most natural adaptation is to use a voxel-grid representation of the 3D object and amend the 2D CNN framework to use collections of 3D filters for cascaded processing in the place of conventional 2D filters. A number of problems arise from this approach. Firstly, for a fixed cost there is a tremendous discrepancy in the obtainable resolution of the input. Processing a 30x30x30 voxel grid is comparable in computational cost to processing a 164x164 image but the relative detail inherent in these representations is immensely different. Secondly, the space of

deformations in 3D is much larger than in 2D. This requires networks operating on 3D inputs to afford greater capacity. Greater capacity, however, requires larger training sets and while the prevalence of 3D models is increasing it is still much less than their 2D counterparts.

Several attempts have been made to use CNNs to produce discriminative representations from volumetric data. ShapeNets [10] propose a fully-volumetric network with three 3D convolutional layers followed by two fully-connected layers. Qi et al. [25] observe significant overfitting when attempting to train the aforementioned end-to-end and choose to amend the technique using subvolume classification as an auxiliary task. Qi et al. [25] also propose an alternate 3D CNN which learns to project the volumetric representation to a 2D representation which is then processed using a conventional 2D CNN architecture. Even with these adaptations, Qi et al. [25] are challenged by overfitting and suggest augmentation in the form of orientation pooling as a remedy. VoxNet [21] developed concurrently with ShapeNet [10] is a volumetric CNN with two instead of three 3D convolutional layers and two fully connected layers. VoxNet [21] incorporates a multiresolution approach by concatenating the representations of 3D CNNs operating on inputs of different resolution. This approach gives VoxNet [21] a slight improvement when compared to their single resolution approach.

Qi ei al. [24] also present an attempt to train a neural network that operates directly on point clouds.

Currently, the most successful approaches are view-based, operating in rendered views of the 3D object [25, 16, 2]. The high performance of these methods is in part due to the use of large pre-trained 2D CNNs (on ImageNet, for instance).

Recently, a body of work on Graph Convolutional Networks (GCN) has emerged. There are two threads within this space, spectral [6, 8, 17] and spatial [5, 20, 22]. These approaches learn filters on irregular but structured graph representations. These methods differ from ours in that we are looking to explicitly learn equivariant and invariant representations for 3D shapes under rotation. While such properties are difficult to construct for general manifolds, we leverage the group action of rotations on the sphere.

3. Mathematical Preliminaries

Consideration of symmetries, in particular rotational symmetries, naturally evokes notions of the Fourier Transform. In the context of deriving rotationally invariant representations, the Fourier Transform is particularly appealing since it exhibits invariance to rotational deformations up to phase (a truly invariant representation can be achieved through application of the modulus operator).

To take advantage of this property for 3D shape representation, it is necessary to construct a rotationally equivariant representation of our 3D input. For a group G and function $f : E \to F$, f is said to be equivariant to transformations $g \in G$ when

$$f(g \circ x) = g' \circ f(x), \quad x \in E \tag{1}$$

where g acts on elements of E and g' is the corresponding group action which transforms elements of F. If $E = F$, $g = g'$. A straightforward example of an equivariant representation is an orbit. For an object x, its orbit $O(x)$ with respect to the group G is defined

$$O(x) = \{g \circ x \mid \forall g \in G\}. \tag{2}$$

Through this example it is possible to develop an intuition into the equivariance of the group convolution; convolution can be viewed as the inner-products of some function f with all elements of the orbit of a "flipped" filter h. Formally, the group convolution is defined

$$(f \star_G h)(x) = \int_{g \in G} f(g) h(g^{-1}x)\, dg. \tag{3}$$

The familiar classical convolution is a special case of the group convolution with the group $G = \mathbb{R}^n$,

$$\begin{aligned}(f \star h)(x) &= \int_{g \in \mathbb{R}^n} f(g) h(g^{-1}x)\, dg \\ &= \int_{g \in \mathbb{R}^n} f(g) h(x - g)\, dg.\end{aligned} \tag{4}$$

The group convolution can be shown to be equivariant by

$$\begin{aligned}((\alpha^{-1} f) \star_G h)(x) &= \int_{g \in G} f(\alpha^{-1} g) h(g^{-1}x)\, dg \\ &= \int_{\tilde{g} \in G} f(\tilde{g}) h((\alpha\tilde{g})^{-1}x)\, d\tilde{g} \\ &= \int_{\tilde{g} \in G} f(\tilde{g}) h(\tilde{g}^{-1}\alpha^{-1}x)\, d\tilde{g} \\ &= (f \star_G h)(\alpha^{-1}x) \\ &= \alpha^{-1}((f \star_G h)(x)).\end{aligned} \tag{5}$$

Given a spherical function f representative of the 3D model, and h, a spherical filter, their convolution with respect to the 3D rotation group $\mathbf{SO}(3)$ is a function $\tilde{f}$ on the sphere. The evaluation of the convolution integral requires some consideration.

As proposed in Section 5 of [15], it is desirable to choose a spherical discretization which has regular, compact cells of the same shape and size. These cells should be small enough to provide good angular resolution and of a shape such that some rotation brings the cells into coincidence. The discussion in [15] makes it plain that there is no such

discretization. The simple and familiar discretization by latitude and longitude suffers from inconsistent cell shape and the platonic solids which satisfy many of the proposed characteristics do not offer cells small enough to provide good angular resolution. Even if a satisfactory discretization were identified, a system of cascaded convolutions on the sphere quickly becomes prohibitively expensive.

Fortunately, the machinery of Fourier analysis extends to functions on the sphere through the spherical harmonics, and it is possible to circumvent convolution in the spatial domain by opting to perform multiplication in the frequency domain instead. Similarly to the harmonics used in computation of the classical Fourier Transform (i.e. the solutions of the spherical Laplacian on $\mathbf{S}^1$, $\{e^{in\theta} \,|\, n \in \mathbb{Z}\}$), the spherical harmonics are the eigenfunctions of the spherical Laplacian on $\mathbf{S}^2$.

The utilization of spherical harmonics in representing, convolving and correlating spherical functions in the Spherical CNN framework is detailed in Section 4. A more comprehensive exposition of the spherical harmonics can be found in [12].

4. Implementation

Our model first step is to convert a 3D object to a spherical function, which is discussed in 4.3. Then, we run a sequence of blocks containing spherical convolutions, which are described in 4.1. Section 4.2 details the architecture design choices. Figure 2 shows an overview of our method.

4.1. Spherical convolutions

Any bandlimited function f on the sphere can be finitely expanded into the spherical harmonics basis [1]:

$$f = \sum_{0 \le l \le b} \sum_{|m| \le l} \hat{f}_m^l Y_m^l, \tag{6}$$

$$\hat{f}_m^l = \int_{S^2} f(x) \overline{Y_m^l} dx, \tag{7}$$

where b is the bandwidth, and Y_m^l is the spherical harmonics of degree l and order m. We refer to (7) as the Spherical Fourier Transform (SFT), and to (6) as its inverse (ISFT). The sampling theorem on the sphere [9] allows computation of $\hat{f}_m^l$ from equiangular samples of f ,

$$\hat{f}_m^l = \frac{\sqrt{2\pi}}{2b} \sum_{j=0}^{2b-1} \sum_{k=0}^{2b-1} a_j^{(b)} f(\theta_j, \phi_k) \overline{Y_m^l}(\theta_j, \phi_k), \tag{8}$$

$$a_j^{(b)} = \frac{\sqrt{2}}{b} \sin\left(\frac{\pi j}{2b}\right) \sum_{l=0}^{b-1} \frac{\sin\left([2l+1]\frac{\pi j}{2b}\right)}{2l+1}, \tag{9}$$

where $\theta_j = \pi j / 2b$ and $\phi_k = \pi k / b$ form the sampling grid.

Let $y = f * h$ denote the spherical convolution between f and h. By the convolution theorem on the sphere [9],

$$\hat{y}_m^l = 2\pi \sqrt{\frac{4\pi}{2l+1}} \hat{f}_m^l \hat{h}_0^l, \tag{10}$$

which gives a direct way to compute the spherical convolution:

1. expand f and h into the spherical harmonics basis (8),
2. compute weighted pointwise multiplication of the coefficients (10), and
3. invert the spherical harmonics expansion (6).

Note that all the required operations are matrix pointwise multiplications and sums, which are differentiable and readily available in most automatic differentiation frameworks. In our direct implementation, we precompute all needed Y_m^l, which are stored as constants in the computational graph.

We also implement a potentially faster method based on separation of variables, shown by [9]. Expanding Y_m^l in (8), we obtain

$$\begin{aligned} \hat{f}_m^l &= \sum_{j=0}^{2b-1} \sum_{k=0}^{2b-1} a_j^{(b)} f(\theta_j, \phi_k) q_m^l P_m^l(\cos\theta_j) e^{-im\phi_k} \\ &= q_m^l \sum_{j=0}^{2b-1} a_j^{(b)} P_m^l(\cos\theta_j) \sum_{k=0}^{2b-1} f(\theta_j, \phi_k) e^{-im\phi_k}, \end{aligned} \tag{11}$$

where P_m^l is the associated Legendre polynomial, and q_m^l a normalization factor. The inner sum can be computed using a row-wise Fast Fourier Transform, which is implemented in some automatic differentiation frameworks (e.g., Tensorflow), and what remains is an associated Legendre transform, which we compute directly. We found that this method is faster for $b \ge 32$, but in most experiments we keep $b \le 16$, and use the direct method. Note that there are faster algorithms available [9, 14], which we did not attempt to implement.

4.1.1 Spectral filtering

The network is supposed to learn useful representations by learning the filters that are convolved with the inputs and feature maps.

Here we define the filter parameterization. A natural way would be to define a compact support around one of the poles and learn the values for each discrete location, setting the rest to zero. The downside of this approach is that there are no guarantees that the filter will be bandlimited. If it is not, the SFT will be implicitly bandlimiting the signal, which causes a discrepancy between the parameters and the actual realization of the filters.

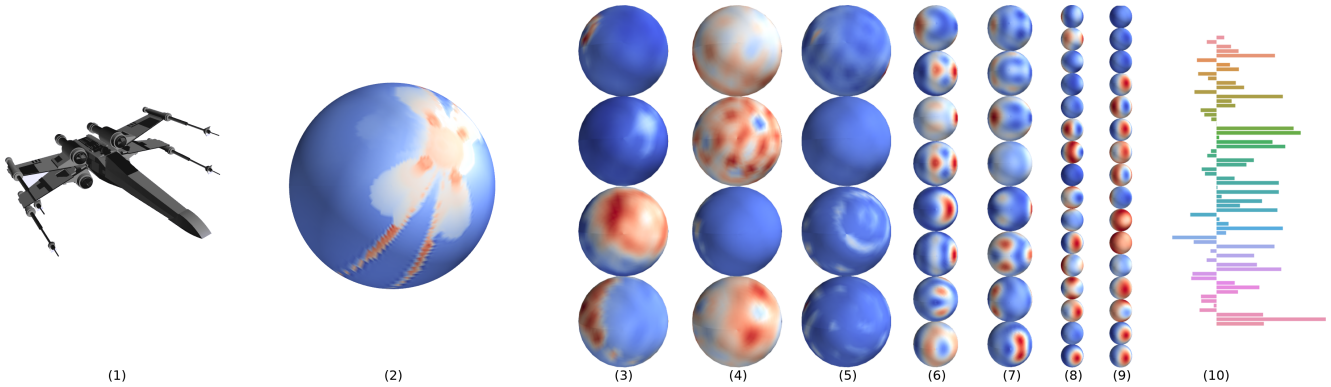

Figure 2. Overview of our method. From left to right: a 3D model (1) is mapped to a spherical function (2), which passes through a sequence of spherical convolutions, nonlinearities and pooling, resulting in equivariant feature maps (3–9). We show only a few channels per layer. A global weighted average pooling of the last feature map results in a descriptor invariant to rotation (10), which can be used for classification or retrieval. The input spherical function (2) may have multiple channels, in this picture we show the distance to intersection representation.

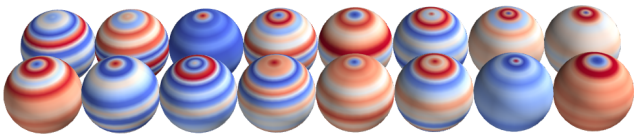

Figure 3. Filters learned by the first spherical convolutional layer. The filters are zonal. These are parameterized by 16 spectral coefficients, and we show the correspondent 32×32 spherical functions. Even though locality is not enforced, some filters learn to respond locally.

We found that a superior approach is to learn the filters in the spectral domain. First note that, to compute the convolution of a function f and a filter h, only the SFT coefficients of order $m = 0$ of h are used. In the spatial domain, this implies that for any h, there is always a zonal filter (constant value per latitude) h_z, such that $\forall y,\ y * h = y * h_z$. Thus, it only makes sense to learn zonal filters, and we parameterize them by the SFT coefficients of order $m = 0$. For a 32×32 sampling, the maximum workable bandwidth is $b = 16$, so there are 16 parameters to be learned $(\hat{h}_0^0, \dots \hat{h}_0^{15})$.

The spectral parameterization is also faster because it eliminates the need to compute the filter SFT, since the filters are defined in the spectral domain, which is the same domain where the convolution computed. The downside is that the filters are not necessarily local; however, the locality may be learned. Figure 3 shows some filters learned by our model.

4.2. Architecture

We define a block as one spherical convolution layer, followed by optional pooling, and nonlinearity. A weighted global average pooling is applied at the last layer to obtain an invariant descriptor.

4.2.1 Spectral pooling

The conventional spatial max pooling used in CNNs have two drawbacks in Spherical CNNs: (1) need to run an expensive ISFT to convert back to spatial domain, and (2) equivariance is not completely preserved, specially because of unequal cell areas from equiangular sampling. Average pooling could take into account the cell areas to mitigate the latter, but would still be affected by the former. We propose to use spectral pooling. If the input has bandwidth l, we remove all coefficients with degree larger or equal than $l/2$ (effectively, a lowpass filter). Note that spectral pooling was proposed before for conventional CNNs [26].

We experiment with both spectral pooling and weighted average pooling (WAP), and found that there is a trade-off between computation cost and accuracy, where the WAP improves accuracy, but is significantly slower.

4.2.2 Nonlinearity

We noticed that using spherical convolutions and randomly initializing the spectral filters results in several features maps with all negative values for all dataset entries. If ReLU is chosen as the nonlinearity, these would all be set to zero and so would the gradients flowing through them, precluding learning. We workaround this issue by using Parametric ReLU (PReLU) [13] instead, which learns a constant α to multiply negative entries, instead of setting them to zero. We initialize $\alpha = 0.1$.

The nonlinearity is applied in the spatial domain. In fact, this is the only reason to compute the ISFT at every layer. In pooling blocks, the nonlinearity is applied after pooling, so the ISFT is done at lower resolution, hence faster.

One drawback is that we cannot guarantee that the PReLU output is bandlimited, so the feature maps may have

their highest frequencies discarded at the next SFT.

4.2.3 Weighted global average pooling (WGAP)

In fully convolutional networks, it is usual to apply a global average pooling at the last layer to obtain a vector descriptor per input, where each entry is the average of one feature map. We use the same idea; however, the equiangular spherical sampling results in cells of different areas, so we compute a weighted average instead, where a cell's weight is the sine of its latitude. Note that the WGAP is invariant to rotation, therefore the descriptor is also invariant.

An alternative to this approach is to use the magnitude per degree of the SFT coefficients; formally, if the last layer has bandwidth b and $\hat{f}^l = [\hat{f}^l_{-l}, \hat{f}^l_{-l+1}, \dots, \hat{f}^l_l]$, then $d = \left[\left\|\hat{f}^0\right\|, \left\|\hat{f}^1\right\|, \dots \left\|\hat{f}^{b-1}\right\|\right]$ is an invariant descriptor [1]. We experimented briefly with this approach and did not find noticeable performance improvements, so we use the WGAP.

4.2.4 Increasing the input resolution

The spherical convolution is computationally more expensive than a planar convolution, which limits our input resolution. Concretely, we can train a medium sized network on ModelNet40 in a couple hours for an input resolution of 32×32, which implies a maximum bandwidth $b = 16$.

Some performance improvement is observed by increasing the input resolution; however, the training time increases steeply. In order to increase the workable input resolution, and keep the training time manageable, we precompute the input SFT for the whole dataset at a higher resolution and use it as the input. The first convolution is just a pointwise multiplication, since the input and filter are already in the spectral domain. Then, to avoid a costly ISFT, we immediately perform spectral pooling to a manageable resolution. Our best results for single-branched networks are with an input resolution of 128×128, pooled to 32×32 after the first convolution.

4.3. Spherical 3D object representation

The problems we are interested on take 3D objects as inputs, which are usually represented by a mesh or voxel grid. We discuss how to convert from these representations to functions on the sphere. Note that the conversion function itself must be equivariant to rotations; our learned representation will not be equivariant if the input is pre-processed by a non-equivariant function.

4.3.1 Ray-mesh intersection

Given a mesh or voxel grid, we first find the bounding sphere and its center. Given a desired resolution n, we cast $n \times n$ equiangular rays from the center, and obtain the intersections between each ray and the mesh/voxel grid. Let d be the distance from the center to the farthest point of intersection, for a ray at direction (θ_j, ϕ_k). We create the function on the sphere by making $f(\theta_j, \phi_k) = d$, $0 \leq j, k \leq n$.

In case the input is a mesh, we also compute the angle α between the ray and the surface normal at the intersecting face, and include it as a second channel $f(\theta_j, \phi_k) = [d, \sin \alpha]$.

Note that this representation is suitable for star-shaped objects, defined as objects that contain an interior point from where the whole boundary is visible. Moreover, the center of the bounding sphere must be one of such points. In practice, we do not check if these conditions hold – even if the representation is ambiguous or non-invertible, it may still be useful.

4.3.2 Other representations

We experiment with several alternative representations, but none turned out better than the ray-mesh intersection.

- *multichannel*, where each channel corresponds to a concentric sphere, and d is computed considering only intersections within its sphere, potentially relaxing the star-shaped object restriction;
- *complex representation*, where $f(\theta_j, \phi_k) = de^{i\alpha}$, which allows the use of complex filters;
- *signed distance functions*, where d is replaced by the distance from the surface of the sphere to the object, ignoring the intersections; and also its multichannel variant, where the distance is made negative if the point is inside the object; and
- *radial embedding*, where we sample a binary vector along each ray indicating if each point is inside or outside the object, then learn an embedding from this vector to a lower-dimensional one.

5. Experiments

We experiment with variations of a six layer base model, with pooling after the second and fourth layers. The best performance was obtained with the largest we could train: [32, 32, 64, 64, 128, 128] channels per layer. When using spectral pooling, an extra layer is included upfront, with the same number of channels as the first.

Our best results use the two-channel (distance, angle) representation (4.3.1). The two channels are split in two branches, and intermediate features from the angle branch are concatenated with the distance branch before each pooling layer, with a total of 548k parameters.

	micro			macro				
	P@N	R@N	mAP	P@N	R@N	mAP	input size	params
Furuya [11]	0.814	0.683	0.656	0.607	0.539	0.476	$100 \times 7 \times 6 \times 3 \times 10$	8.4M
Tatsuma [30]	0.705	0.769	0.696	0.424	0.563	0.418	38×224^2	3M
Ours	0.690	0.684	0.630	0.439	0.490	0.408	2×32^2	0.5M
Zhou [2]	0.660	0.650	0.567	0.443	0.508	0.406	50×224^2	36M

Table 1. Results for the SHREC'17 contest, perturbed dataset. We compare precision, recall and mean average precision (mAP) for the top 3 teams. *micro* indicates averaging adjusted by category size, *macro* has no such adjustment. The mAP metric is used for ranking, which would put our model in 3rd place. Our performance is competitive even with significantly fewer parameters, smaller input resolution, and no pre-training.

We train the models for 32 epochs, with Adam and initial learning rate of 10^{-3}, dividing it by 10 half-way through training.

We make extensive use of data augmentation, performing rotations, anisotropic scaling and mirroring on the meshes, and adding jitter to the bounding sphere center when constructing the spherical function. Note that, even though our learned representation is equivariant to rotations, augmenting the inputs with rotations is still beneficial due to interpolation and sampling effects.

The greatest advantage of our model is inherent equivariance to SO(3) rotations; we focus the experiments in problems that benefit from it, but also show results in other problems.

5.1. 3D object retrieval

We run shape retrieval experiments on ShapeNet Core55 [7], following the rules of the SHREC'17 3D shape retrieval contest [28], which includes a dataset perturbed with random SO(3) rotations.

The network is trained for classification on the 55 core classes (we do not use the subclasses), and the invariant descriptor is used with a cosine distance for retrieval. Given a query, we return all elements from the same predicted class. Table 5 shows the results. The top ranked teams either used pre-trained 2D-CNN on rendered views [30, 2], or handcrafted features [11], all with several million parameters. Our model achieves competitive performance, with significantly fewer parameters and smaller input size, with no need for pre-training.

Our invariant descriptors result in rotation-agnostic retrieval. To show that, we run a retrieval experiment on ModelNet40, perturbed with SO(3) rotations. At most 3 rotated versions of the same instance are allowed in the retrieval test set; since their descriptors should be almost equal, the models should appear clustered in the retrieval list. Figure 4 shows the results.

5.2. 3D object classification

We study the effects of SO(3) rotations in classification performance using ModelNet40 [31]. We perturb both the ModelNet40 training and test sets with random SO(3) rotations, and compare with VoxNet [21], a voxel based approach, which is retrained for the perturbed dataset. Table 5.2 shows the results. VoxNet performance drops sharply with SO(3) rotations. Our model is more robust, but the performance still decline. We attribute this to (1) class ambiguity introduced by SO(3) rotations, and (2) sampling effecs (see 5.3 for discussion).

Model	z-rotations	SO(3) rotations	# params
Ours	84.6%	81.3%	548k
Ours (small)	84.0%	79.8%	140k
VoxNet [21]	83%	73%	920k

Table 2. Average accuracy per class on ModelNet40 with z-rotations and SO(3) rotations. VoxNet performance drops sharply on the larger space of rotations, while our model is more robust.

Most results on ModelNet40 are given for upright models with arbitrary azimuthal rotations (z-rotations). Equivariance to SO(3) rotations is not necessary in this case; although equivariance to z-rotations is desirable; however, conventional approaches seem to be able to implicitly handle it. The full potential of our model is not exercised on this dataset; we present results on it for completion. Table 5.2 shows the results. Our model performance is superior to the voxel-based approaches, but inferior to the multi-view and point-cloud based. Regarding number of parameters and input size, our model is significantly smaller than the others.

Model	p. class	p. inst.	params	inp. size
MVCNN-MultiRes [25]	91.4%	93.8%	180M	60×224^2
PointNet [24]	86.2%	89.2%	3.5M	1024×3
Ours	84.6%	86.9%	0.5M	2×32^2
VoxNet [21]	83%	-	0.9M	30^3
3D ShapeNets [31]	77.3%	-	11M	30^3

Table 3. Classification on ModelNet40 with z-rotations. We compare accuracies per class and per instance with multi-view, point-cloud, and voxel based approaches. Our model uses significantly fewer parameters and smaller input sizes, but shows only average performance for this dataset, since the z-rotations are more easily handled by conventional approaches.

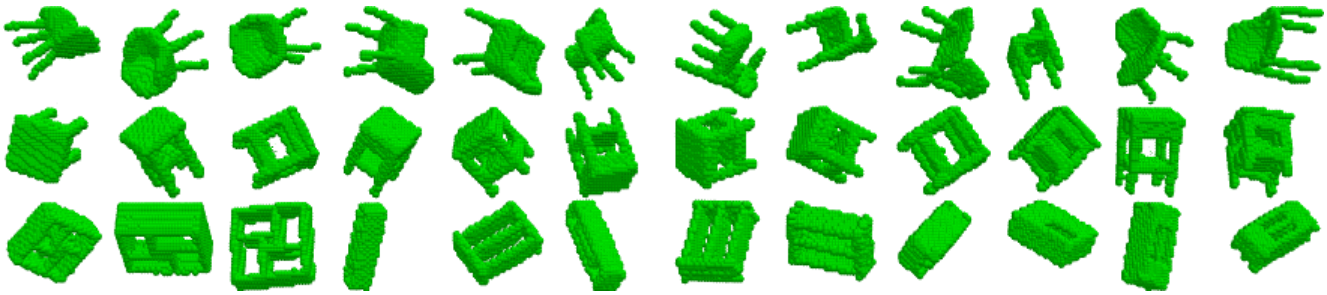

Figure 4. Retrieval on ModelNet40 perturbed with SO(3) rotations. Left-most column shows the query, others are retrieved objects, sorted by distance to query. Due to the invariant descriptor, our model is rotation agnostic. The retrieval set contains at most 3 rotated versions of the same instance; notice how they often appear clustered in the retrieved list.

5.3. Visualization

Figure 5 shows some rotated inputs and corresponding descriptors given by our model. Note that the invariance to SO(3) rotations is only approximate, because the spherical functions constructed from the meshes are not guaranteed to be bandlimited, causing different samplings to produce different outputs. This effect is mitigated for rotations around z, because, since we use equiangular sampling, the cell area varies with the latitude, and rotations around z preserve latitude.

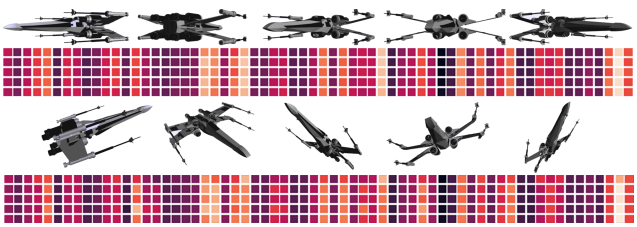

Figure 5. Our model learns descriptors that are invariant to input rotations. From top to bottom: rotations around z and correspondent descriptors (one per row), random SO(3) rotations and correspondent descriptors. The invariance is approximate for arbitrary SO(3) rotations because of sampling effects.

6. Conclusion

We presented the Spherical CNNs, a network that leverages spherical convolutions to achieve equivariance to SO(3) inputs. The network is applied to 3D object classification and retrieval, but has potential applications in spherical images such as panoramas, or any kind of data that can be represented as a function on a sphere. We show that our model can naturally handle arbitrary input orientations for different tasks, requiring relatively few parameters and small input sizes.

References

[1] G. Arfken. *Mathematical Methods for Physicists*. Number v. 2 in Mathematical Methods for Physicists. Academic Press, 1966. 4, 6

[2] S. Bai, X. Bai, Z. Zhou, Z. Zhang, and L. Jan Latecki. Gift: A real-time and scalable 3d shape search engine. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5023–5032, 2016. 3, 7

[3] M. Belkin, J. Sun, and Y. Wang. Discrete laplace operator on meshed surfaces. In *Proceedings of the 24th ACM Symposium on Computational Geometry, College Park, MD, USA, June 9-11, 2008*, pages 278–287, 2008. 2

[4] S. J. Belongie, J. Malik, and J. Puzicha. Shape context: A new descriptor for shape matching and object recognition. In *Advances in Neural Information Processing Systems 13, Papers from Neural Information Processing Systems (NIPS) 2000, Denver, CO, USA*, pages 831–837, 2000. 2

[5] D. Boscaini, J. Masci, E. Rodolà, and M. Bronstein. Learning shape correspondence with anisotropic convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 3189–3197, 2016. 3

[6] J. Bruna, W. Zaremba, A. Szlam, and Y. LeCun. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*, 2013. 3

[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. *CoRR*, 2015. 2, 7

[8] M. Defferrard, X. Bresson, and P. Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852, 2016. 3

[9] J. R. Driscoll and D. M. Healy. Computing fourier transforms and convolutions on the 2-sphere. *Advances in applied mathematics*, 15(2):202–250, 1994. 4

[10] H. Fan, Y. Zhang, Z. Hua, J. Li, T. Sun, and M. Ren. Shapenets: Image representation based on the shape. In *2016 IEEE 14th Intl Conf on Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress,*

DASC/PiCom/DataCom/CyberSciTech 2016, Auckland, New Zealand, August 8-12, 2016, pages 196–201, 2016. 3

[11] T. Furuya and R. Ohbuchi. Deep aggregation of local 3d geometric features for 3d model retrieval. In *BMVC*, 2016. 7

[12] J. Gallier. Notes on differential geometry and lie groups. 2015. 4

[13] K. He, X. Zhang, S. Ren, and J. Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *CoRR*, 2015. 5

[14] D. M. Healy, D. N. Rockmore, P. J. Kostelec, and S. Moore. Ffts for the 2-sphere-improvements and variations. *Journal of Fourier analysis and applications*, 9(4):341–385, 2003. 4

[15] B. K. P. Horn. Extended gaussian images. *Proceedings of the IEEE*, 72(12):1671–1686, 1984. 2, 3

[16] A. Kanezaki, Y. Matsushita, and Y. Nishida. Rotationnet: Joint learning of object classification and viewpoint estimation using unaligned 3d object dataset. *CoRR*, 2016. 3

[17] T. N. Kipf and M. Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016. 3

[18] I. Kokkinos, M. M. Bronstein, R. Litman, and A. M. Bronstein. Intrinsic shape context descriptors for deformable shapes. In *2012 IEEE Conference on Computer Vision and Pattern Recognition, Providence, RI, USA, June 16-21, 2012*, pages 159–166, 2012. 2

[19] A. Makadia and K. Daniilidis. Spherical correlation of visual representations for 3d model retrieval. *International Journal of Computer Vision*, 89(2):193–210, 2010. 2

[20] J. Masci, D. Boscaini, M. Bronstein, and P. Vandergheynst. Geodesic convolutional neural networks on riemannian manifolds. In *Proceedings of the IEEE international conference on computer vision workshops*, pages 37–45, 2015. 3

[21] D. Maturana and S. Scherer. Voxnet: A 3d convolutional neural network for real-time object recognition. In *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2015, Hamburg, Germany, September 28 - October 2, 2015*, pages 922–928, 2015. 3, 7

[22] F. Monti, D. Boscaini, J. Masci, E. Rodolà, J. Svoboda, and M. M. Bronstein. Geometric deep learning on graphs and manifolds using mixture model cnns. *arXiv preprint arXiv:1611.08402*, 2016. 3

[23] R. Osada, T. A. Funkhouser, B. Chazelle, and D. P. Dobkin. Shape distributions. *ACM Trans. Graph.*, 21(4):807–832, 2002. 2

[24] C. R. Qi, H. Su, K. Mo, and L. J. Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. *CoRR*, 2016. 3, 7

[25] 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 *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 5648–5656, 2016. 3, 7

[26] O. Rippel, J. Snoek, and R. P. Adams. Spectral representations for convolutional neural networks. *CoRR*, 2015. 5

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. S. Bernstein, A. C. Berg, and F. Li. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, 2015. 1

[28] M. Savva, F. Yu, H. Su, A. Kanezaki, T. Furuya, R. Ohbuchi, Z. Zhou, R. Yu, S. Bai, X. Bai, M. Aono, A. Tatsuma, S. Thermos, A. Axenopoulos, G. T. Papadopoulos, P. Daras, X. Deng, Z. Lian, B. Li, H. Johan, Y. Lu, and S. Mk. Shrec'17 track: Large-scale 3d shape retrieval from shapenet core55. In *10th Eurographics workshop on 3D Object retrieval*, pages 1–11, 2017. 7

[29] J. Sun, M. Ovsjanikov, and L. J. Guibas. A concise and provably informative multi-scale signature based on heat diffusion. *Comput. Graph. Forum*, 28(5):1383–1392, 2009. 2

[30] A. Tatsuma and M. Aono. Multi-fourier spectra descriptor and augmentation with spectral clustering for 3d shape retrieval. *The Visual Computer*, 25(8):785–804, 2009. 7

[31] 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 Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 1912–1920, 2015. 1, 2, 7