Novel Shape Generation with SO(3)-Equivariant Auto-Encoders

Shayan Pardis[*]
Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
shayanp@mit.edu

Ethan Chun[*]
Electrical Engineering and Computer Science
Massachusetts Institute of Technology
Cambridge, MA 02139
elchun@mit.edu

Abstract

Novel shape generation is a critical procedure across fields spanning computer graphics to robotics. Unfortunately, current methods for generating 3D shapes are data inefficient due to their inability to utilize the inherent SO(3) symmetry in 3D data. Therefore, we propose a simple, SO(3) equivariant, auto-encoding neural network to test the feasibility of building larger SO(3) equivariant shape generation models. In order to capture the SO(3) symmetry of our input data, we represent 3D shapes as the coefficients of the spherical harmonic basis functions. Then, to perform SO(3) equivariant latent space traversal and produce coherent outputs, we design a rotation equivariant interpolation method that distinguishes between the rotation and shape components of the latent space. We demonstrate robust reconstruction of a dataset of 100 randomly generated boxes and show coherent traversals of the latent space when trained on two and 100 randomly selected objects.

1 Introduction

Automated 3D object generation has the power to revolutionize fields spanning 3D content creation, virtual reality, and robotics. However, current methods are limited in speed and computational efficiency. A large factor in this inefficiency is the diverse set of SO(3) poses that 3D objects can take. Most methods acting on data types such as point clouds, meshes, voxel grids must use data augmentation to ensure that objects are generated in both diverse shapes and poses, increasing the size of the input space that a network must learn. Furthermore, work in generating implicit representations such as neural radiance fields often struggle with consistency between different orientations of the same object.

Spherical harmonic representations of 3D shapes have the potential to alleviate this issues. Specifically, an arbitrary star-shaped signal on the sphere can be encoded to an arbitrarily high degree of precision by the coefficients of the spherical harmonics up to some desired l. This representation has the benefit of being SO(3) equivariant, mirroring real-world objects. SE(3) equivariance can be trivially added by mean centering an object to be encoded.

Leveraging the spherical harmonic datatype, we present a novel, SO(3) equivariant, auto-encoding architecture to test the feasibility of generating 3D shapes using equivariant architectures and the spherical harmonic basis. Furthermore, we develop an SO(3) equivariant interpolation strategy to effectively separate the rotation and shape components of the latent space. To simplify the problem, we run reconstruction and interpolation on a dataset of 100 boxes and parameterize these by the coefficients of the first l spherical harmonics. A graphical illustration of our problem setup can be found in Fig 1. Through our experiments, we prove that our architecture is approximately SO(3)

[*]equal contribution

equivariant, demonstrate robust reconstruction of the boxes dataset, and show novel object generation with our equivariant latent space interpolation. We also provide commentary on the training dynamics of the architecture and the effect of the latent space dimension.

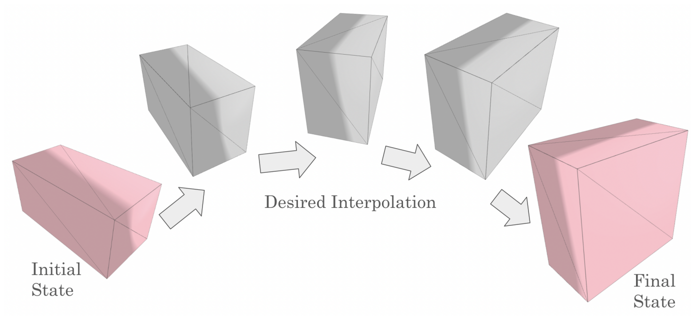

Figure 1: **Problem Overview** – As a first step in building SO(3) equivariant 3D generative models, we seek to demonstrate equivariant interpolation between objects from a dataset of boxes. This interpolation should distinguish between the rotation and shape components of the objects, acting as if the objects were rotated to a canonical form, linearly interpolated, then rotated back. The canonical form should be learned by the network.

2 Related Work

2.1 3D Shape Generation

The field of 3D shape generation has seen substantial interest in recent years. Works can be split into the data type produced and the method to used to generate the data. Some of the earliest models are those based on Generative Adversarial Networks [1] such as Henzler, Mitra, and Ritschel's PlatonicGAN [2]. This method uses a generator to generate an object represented on a voxel grid, which is then rendered and passed into a discriminator. In recent time, pure voxel grids have fallen out of favor due to high memory consumption and correspondingly low resolution. Similarly, GANs have largely been superseded by diffusion based models [3] due to the poor training characteristics exhibited by GANs.

Following the release of Neural Radiance Fields (NeRF) [4], a number of methods have focused on generating data generated by implicit fields. Implicit fields have the benefit of higher resolution for a given memory amount and can model lighting and geometric properties. Schwarz et al. introduce GRAF [5] which produce radiance fields from 2D images using a GAN style architecture. PiGan [6] also produces radiance field outputs but builds on the related work of Sitzmann et al. on Implicit Neural Representations with Periodic Activation Functions [7]. Schwarz et al. build on GRAF with VoxGRAF [5], parameterizing the radiance field as a sparse voxel grid for higher resolution synthesis. While these methods produce high fidelity visual results, training and inference is often computationally intensive due to the need to render the radiance fields into an image for evaluation and back propagation. This limitation can be partially remedied by switching to voxel based implicit representations [8] or to Gaussian Splatting methods [9].

Concurrent with the release of NeRF was the introduction of Diffusion Models for image synthesis [3]. This spurred a number of diffusion based 3D shape generators. These models commonly work by synthesizing multiple views of the same object, given a reference image or input prompt. These views can then be used to train a neural radiance field. Among these are DreamFusion [10], HoloDiffusion [11], and MVDream [12]. Changing data types from 2D images to point clouds, Lou and Hu use a diffusion model to directly generate point clouds [13]. A large challenge with image based diffusion models is devising methods to ensure view consistency between multiple generated views.

To the authors' knowledge, no work uses deep learning to produce any 3D objects parameterized by the spherical harmonic basis.

2.2 SO(3) Equivariant Architectures

As discussed previously, a large number of models are plagued by difficulties concerning SO(3) equivariance and multi-view consistency. Rather than building equivariance through data augmentation, recent work on SO(3) equivariant architectures draws from the group theoretical foundations of the SO(3) rotation group to build network architectures that are equivariant to rotations in 3D space. The earliest works in the field focused on building spherical CNNs as presented by Cohen et al [14] and Esteves et al. [15]. Clebsch-Gordan Nets [16] extend this by using the Clebsch-Gordan transform in the spherical harmonics basis and thereby avoided the repeated Fourier transformations used in Spherical CNNs. Tensor Field Networks [17] introduced an architecture that works well for point clouds. By using point convolution and message-passing, this architecture prevents the need to approximate the input by a grid which reduces the quality. Subsequent works expanded the set of equivariant features. These include 3D Steerable CNNs [18] and SE(3) Transformers [19]. More recently, Vector Neurons [20] design equivariant layers as a plug-in replacement of layers in otherwise non-equivariant networks.

Due to the exploratory nature of this work, we most closely base our model on the simplest variations of these architectures, namely Spherical CNNs. We also discuss empirical training dynamics of our Spherical CNN based approach in comparison to a Clebsch-Gordan style network.

2.3 SO(3) Equivariant Generative Models

While no current works demonstrate generation of 3D shapes, a number of works use application specific SO(3) equivariant models to generate data through an autoencoding style model. Most closely related to our work is the work by Costa et al. on Protein modeling [21]. This work focuses on modeling protein structures with SO(3) equivariant autoencoders and build a latent diffusion model to generate these proteins. Our work differs by tailoring our architecture to the domain of 3D shape generation as opposed to atomic systems.

3 Background

This work requires a basic understanding of group theory with a special emphasis on infinite groups. In particular, we represent objects as functions on the sphere parameterized by in the spherical harmonic basis of SO(3). Correspondingly, we design our architecture to be SO(3) equivariant.

3.1 Rotation Groups

A group, G, is defined as a collection of elements such that for all $g, h \in G$, $gh \in G$. Furthermore, elements are associative, there exists an identity element e, and all elements have an inverse such that $gg^{-1} = e$. Among many applications, groups can be used to represent symmetries in space. We are particularly interested in rotation equivariance. Therefore, we pay special attention to the group of rotations in n dimensional space, the special orthogonal group SO(n). In terms of linear algebra, this group can be represented by the collection of all distance preserving (orthogonal) $n \times n$ matrices that preserve a fixed point and have a determinant of 1. When $n = 3$, this infinite group, SO(3), contains all possible rotations in 3D space.

3.2 Spherical Harmonics

Critical to our work is the representation of 3D data. In contrast to prior works, we represent objects in the spherical harmonic basis. The spherical harmonics are an infinite set of functions that form a basis for functions on the surface of the sphere (on S2). They are referred to by indices l and m where $l \in \mathbb{N}$ and $-l \leq m \leq l$ for a given pairing of l and m, Y^l_m. Given an arbitrary signal on S2, one can uniquely decompose this signal into a set of coefficients for the spherical harmonics up to some l_{max} where a higher l_{max} results in a higher resolution representation as shown:

$$\hat{f}^l_m = \int_{S^2} f(w) Y^l_m(w) dw$$

A closed object A is star-shaped if for all $p \in A$, we have $\alpha \in [0,1] \rightarrow \alpha p \in A$. We can encode any star-shaped object by creating a signal on S2 such that the value of the signal corresponds to the distance of the surface of the object from the center of the sphere. Visually, one can imagine mean centering an object, then casting an infinite number of rays from the origin outwards. The length of the ray when it intersects the object is the value of the function at the location on the unit sphere intersected by the ray. So long as each ray only intersects the object once (the star shaped constraint), any object can be encoded to an arbitrarily high degree of precision.

4 Methods

Our primary aim is to build a system to generate new objects parameterized by the coefficients of the spherical harmonic basis. We accomplish this by training an autoencoder to reconstruct objects parameterized by the coefficients of the spherical harmonics. Furthermore, we construct the network such that the latent space of the network is SO(3) equivariant and, when traversed, produces smooth changes in output. Given a rich enough latent space, we theorize that by sampling the latent space, we will be capable of generating new data consistent with the training data.

Our approach solves three key challenges. The first is the ability to reduce and increase the dimension of the latent space while preserving equivariance. This includes a choice on how the dimension of the latent space is actually defined. The next is a way to traverse the latent space in a way that distinguishes the rotation and shape components of the object. The final challenge is to devise a loss that encourages the network to reconstruct the object. We also include details on how to convert existing mesh based data to the spherical harmonic coefficients.

4.1 Data Representation

As shown in Fig 2, to transform a mesh to a spherical harmonics representation of a signal, we (1) centralize the mesh, (2) view the signal $f : S^2 \rightarrow R$ as the depth of the mesh at each direction which we calculate using ray casting, and (3) numerically compute the following integral to get the coefficients of spherical harmonics. This follows from the method described in [14].

$$\hat{f}^l_m = \int_{S^2} f(w) Y^l_m(w) dw$$

Additionally, we normalize the coefficients using norm normalization. It is worth mentioning that the standard grid on S^2 is not uniform in surface thus one cannot calculate the integral by averaging the values on the grid. However, with appropriate correction, we found that given a dense enough grid, we could still generate reasonable looking spherical harmonic representations of our input meshes.

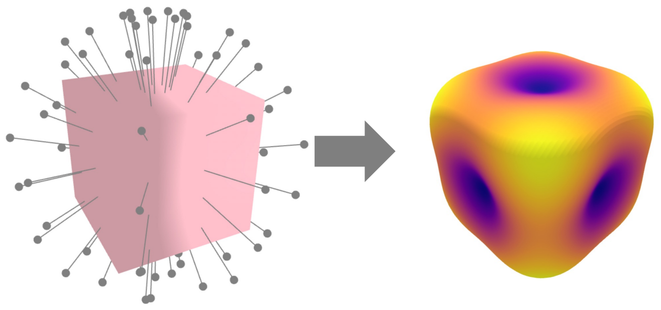

Figure 2: **Mesh to Spherical Harmonics** – We center the mesh, then cast rays along a α, β grid to get the surface distance from the origin. This grid is used to calculate the spherical harmonic coefficients to degree l_{max}. We set $l_{max} = 4$

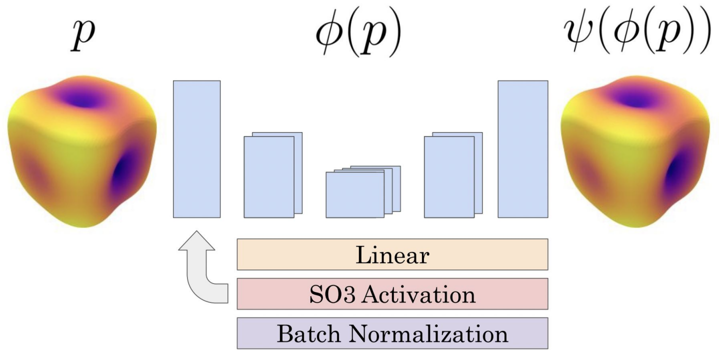

Figure 3: **Spherical CNN Autoencoder Architecture** – As the layers proceed in the encoder, the number of channels of each irrep increases while the degree of irreps decreases. The architecture of the decoder mirrors the encoder.

4.2 Network Architecture

As illustrated in Fig 3, our network consists of an autoencoder on the spherical harmonic coefficients representing the object's shape as an input. When representing the object, we use irreps up to $l_{max} = 4$ in our design. Increasing l_{max} is a trade-off between the quality and the size of the network.

Our first step is to lift our representation from S^2 to $SO(3)$ as the convolution between two spherical signals (e.g. $[f \star h](R)$), lives in $SO(3)$. The tensor product in this representation is a generalized form of convolution [16].

We use a sequence of transformation blocks to change the degree, l_{max}, of irreps and the number of channels. Each block consists of a linear layer, followed by an SO(3) activation layer, followed by batch normalization.

- The linear layer changes the number of channels while preserving the degree of irreps.
$$\{A^{l,k}\}_l \rightarrow \{\sum_k w_k A^{l,k}\}$$
- The SO(3) activation layer changes l_{max}. SO(3) activation evaluates the current representation on SO(3), applies a point-wise non-linearity, and projects the result back to a new representation with a new l_{max}. This can be viewed as an up-sampling (in the decoder) or a down-sampling (in the encoder) mechanism. We use $ReLU$ as the non-linear function, ϕ.
$$\{A^{l,k}\}_l \rightarrow \{\int_{SO(3)} \phi(\sum_{l,k} A^{l,k} \cdot Y^l(x))Y^j(x)dx\}_j$$
- The batch normalization layer is required for training. Our experiments show that without some form of normalization of the norm, the gradients would vanish.

Drawing an analogy to convention CNNs, we use the maximum degree of the spherical harmonic coefficients, l_{max} as the "dimension" of our data and the number of copies of the irreps as the "channels." In the encoder, a sequence of the blocks described above decreases l_{max} while increasing the number of channels c. In the latent representation, we have $l_{max,latent} = 1$ as this is necessary for our interpolation to be equivariant. The decoder increases l_{max} while decreasing c to reproduce the input.

4.2.1 Loss Function

We observe that L_2 distance on the spherical harmonics is not a good metric of the distance of shapes. This intuitively makes sense as the shape of the object is more sensitive to the coefficients of spherical harmonics with smaller l. Thus we define our loss function in the S^2 space.

The point-wise distance of functions in an lp norm is a good candidate:

$$\|f_1 - f_2\|^p = \int_{S^2} (f_1(x) - f_2(x))^p$$

In practice, we implement this idea by sampling a grid over S^2 and computing the point-wise distance:

$$L(f_1, f_2) = \frac{1}{n} \sum_i l(f_1(X_i), f_2(X_i))$$

Thus, our loss function is a random variable which is concentrated around the desired distance metric for large enough n. Using Huber Loss for l empirically made the optimization more robust.

4.3 Interpolation

Given two objects p, q we would like to be able to interpolate between those in a way that disentangles group actions and transformation in the object's canonical form. In another words, we want to find $\gamma_{p,q}(t) : [0, 1] \to \mathcal{D}$ which is an interpolation between p, q (so $\gamma_{p,q}(0) = p, \gamma_{p,q}(1) = q$).

The usual method used in latent space traversal is to linearly interpolate between $\phi(p), \phi(q)$ defined as:

$$\gamma_{,p,q}(t) = \psi(\phi(p) + (1 - t)\phi(q))$$

However, this might not respect the group actions. For instance, in a naive interpolation between rotated versions of the same object might have out-of-distribution in the middle (Fig. 4). This motivates the formalization of equivariant interpolation.

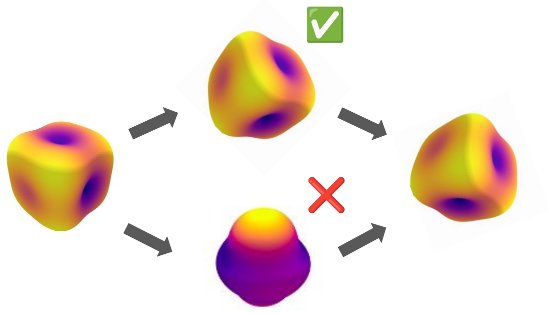

Figure 4: **Pitfalls of Naive Interpolation** – We need equivariant interpolation in order to preserve the shape of objects while rotating.

4.3.1 Equivariant Interpolation

An equivariant interpolation $\gamma_{p,q}$ should have the following properties:

First, a global rotation should commute with an interpolation function. Second, rotating the endpoint q should only rotate the interpolation without changing the shape of the object.

$$\gamma_{g \cdot p, g \cdot q}(t) = D(g)\gamma_{p,q}(t) \tag{1}$$

$$\gamma_{p, g \cdot q}(t) = D(g(\alpha(t)))\gamma_{p,q}(t) \tag{2}$$

The choice of equivariant interpolation is not unique. We designed our interpolation as follows. Let R_v^w be the a rotation between vectors v, w (e.g. $w = R_v^w v$). Then we define:

$$\gamma_{p,q}(t) = \psi\left(\bigoplus_l \left(\frac{\|\phi_l(q)\|}{\|\phi_l(p)\|}\right)^t (R_{\phi_l(p)}^{\phi_l(q)})^t \phi_l(p)\right)$$

In other words, each irrep is rotated independently of the others, and its norm changes as it rotates.

Given that the encoder ϕ and decoder ψ are equivariant, we need the following properties to be satisfied in order to prove our interpolation is equivariant.

$$R_{g\cdot a}^{g\cdot b} = D(g) R_a^b D(g^{-1}) \tag{3}$$

$$R_a^{g\cdot b} = D(g) R_a^b \tag{4}$$

Given this condition, one can verify that the desired properties of an equivariant interpolation will be satisfied (in particular $D(g(\alpha(t))) = D(g)^t$).

However, one can observe that R_a^b is not well defined for $l > 1$. In a d dimensional space, the set of vectors orthogonal to both a, b is $d - 2$ dimensional. Each of the vectors in the orthogonal subspace can be used as the rotation axis and thus for $2l + 1 = d > 3$ the rotation matrix is not unique. This forces our latent space to only $l = 0, 1$ irreps.

5 Experiments

We conduct three experiments. The first is an equivariance test to characterize how equivariant our model is. The next is a reconstruction test to evaluate whether our network is capable of encoding and decoding the shape representations. The last is an interpolation test determine whether our latent space can be used to generate new samples.

5.1 Equivariance Test

Due to grid sampling in the conversion between spherical harmonic and SO(3) space, the model is likely not perfectly equivariant. However, in practical use, small deviations from equivariant are permissible. We are concerned with the equivariance of the encoder and decoder separately as during inference, these modules will be used separately. Given encoder or decoder f, group action g, and input spherical harmonic coefficients x, equivariance is defined as:

$$f(gx) = gf(x)$$

To measure the deviation from equivariance, we compute the mean absolute difference between $f(gx)$ and $gf(x)$ for over all objects in the dataset, D. This is repeated for 10 evenly spaced rotations about the x, y, and z axis.

$$\epsilon(g) = \frac{1}{|D|}\sum_D |f(gx) - gf(x)|$$

As shown in Table 5.1, the mean error is fairly small throughout the rotation sweep. Interestingly, the encoder error is smaller at 180 degrees, likely due to the C_2 rotation symmetry of all our blocks about the cardinal axes. The decoder error about the y axis is also small at 180 degrees, but the other two axes still have higher error values.

5.2 Reconstruction Test

We build a dataset of $n = 100$ cubes with side lengths $a, b, c \sim Uniform(0.5, 2)$. Then, we train our autoencoder with an encoding $l_{max,in} = 4$ and a latent $l_{max,latent}$ of both 1 and 2. We train both for 6000 epochs with each model taking around 80 minutes on a Nvidia Titan X GPU. An epoch is defined here as a single pass through the dataset.

As shown in Fig 5, we found the $l_{max,latent} = 1$ and $l_{max,latent} = 2$ network quickly jumped to an IOU of 84 and 92 respectively, before training more gradually over the remaining epochs. Both networks reached an IOU of around 98 which produces visually similar reconstructed objects. The $l_{max,latent} = 1$ network trained significantly slower than the $l_{max,latent} = 2$ network, which is

Encoder Equivariance Deviation										
	Angle (deg)									
Axis	0	36	72	108	144	180	216	252	288	324
x	0.000	0.044	0.039	0.040	0.043	0.010	0.045	0.040	0.043	0.048
y	0.000	0.031	0.024	0.025	0.032	0.005	0.031	0.025	0.027	0.033
z	0.000	0.018	0.022	0.023	0.028	0.013	0.026	0.030	0.029	0.030
Decoder Equivariance Deviation										
x	0.000	0.026	0.036	0.030	0.014	0.023	0.015	0.030	0.036	0.026
y	0.000	0.016	0.034	0.035	0.016	0.000	0.016	0.034	0.035	0.016
z	0.000	0.014	0.028	0.037	0.038	0.034	0.038	0.037	0.028	0.015

Table 1: **Equivariance Test** – Mean deviation from equivariance for various rotation angles in both the encoder and decoder.

expected due to the increase amount of information that can be stored in the larger latent space. Note that both network eventually reached similar levels of performance, but that the $l_{max,latent} = 2$ network achieved this around epoch 1000, while the $l_{max,latent} = 2$ network required at least 3000 epochs. The $l_{max,latent} = 2$ also showed large dips in performance throughout the training, before quickly recovering.

Empirically, we observed that the initial reconstructions fit a sphere or cylinder to the object. This is somewhat expected as these smoother shapes would be easier to generate with the lower degree spherical harmonic coefficients used in the latent space. However, closer fitting reconstructions need to use the coefficients closer to l_{max}, requiring the network to learn to up-sample the latent space.

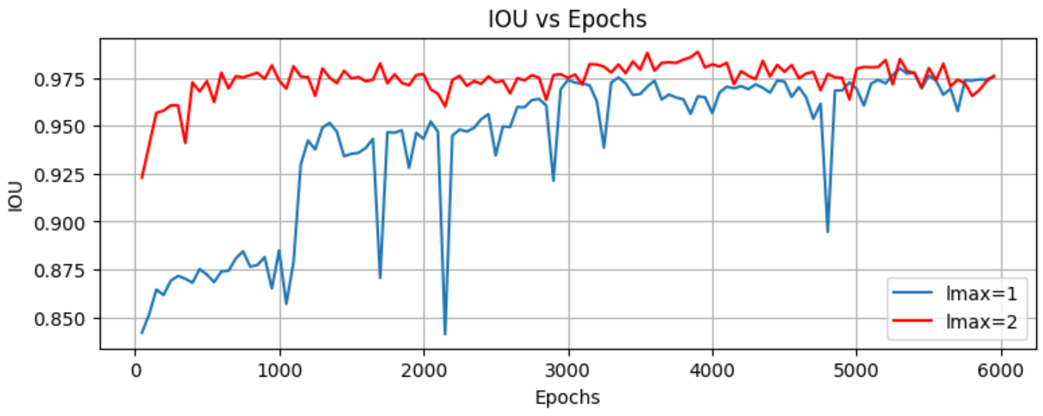

Figure 5: **IOU over training** – We show a sharp initial jump in performance, then more gradual improvements in IOU as training progresses. Note the different rates of convergence for $l_{max} = 1$ and $l_{max} = 2$

5.3 Interpolation Test

We now train our network with an input $l_{max,input} = 4$ and two values of latent $l_{max,latent} = \{1, 2\}$ and two datasets with $n = \{2, 100\}$. Then we experiment interpolating between two objects p, q using both linear interpolation and equivariant interpolation.

As we theoretically justified in section 4.3.1, in $l_{max,latent} = 2$ our interpolation is not equivariant and thus the shape of the object changes dramatically in the middle of the interpolation (Figure 6).

Now we focus on $l_{max,latent} = 1$. In order to emphasize on efficiency of the model, we train on only $n = 2$ objects. The results show that equivariant interpolation transforms between two boxes while staying in the distribution of the boxes (7). In contrast, linear interpolation produces a blob

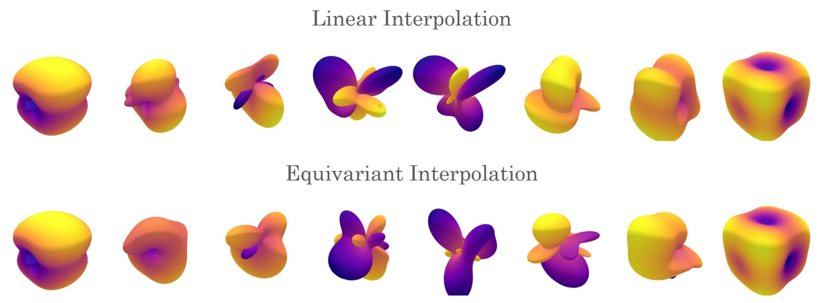

Figure 6: **Interpolation with large latent space and two boxes** – We used $l_{max,latent} = 2$ and trained on $n = 2$ shapes. Neither linear interpolation nor our method is equivariant in the case of $l_{max,latent} = 2$ and the shape significantly changes.

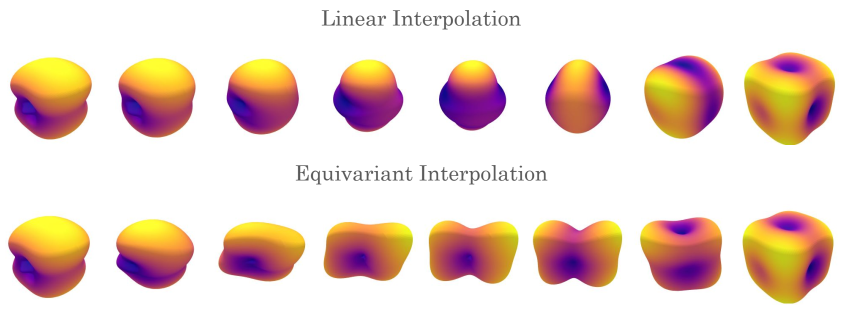

Figure 7: **Interpolation with small latent space and two boxes** – We used $l_{max,latent} = 1$ and trained on $n = 2$ shapes. Equivariant interpolation keeps the shape of a box while interpolating while linear interpolation changes the shape.

in the middle, before returning to the shape. Finally, we trained the model with $n = 100$ (still $l_{max,latent} = 1$) and observed (again) that equivariant interpolation produces a nicer interpolation compared to linear interpolation and the shape stays in the distribution (Figure 8).

Additionally, we observed that there are similarities between linear interpolation and our equivariant interpolation. The reason (as shown in Figure 9) is that linear interpolation approximates traversing the arc while the rotation between the vectors is small. However, with larger rotations, the difference between traversing the arc and the line becomes more significant.

6 Discussion

6.1 Latent Space Dimensionality

As discussed previously, we trained a variant of our model with an $l_{max,latent} = 2$ and one with $l_{max,latent} = 1$. The lower l_{max} model took significantly longer to reach a similar level of performance as the $l_{max} = 2$ model. Unfortunately, we were unable to find a way to perform equivariant traversals in the $l_{max} = 2$ latent space, necessitating the longer training time of the $l_{max} = 1$ model in order to perform latent space exploration. Future work may design equivariant traversal methods for the $l_{max} = 2$ latent space in order to leverage the faster convergence of that model.

6.2 Sampling Methods

In our loss function, we utilize grid sampling to convert from irrep to S^2 space. Any grid defined on the sphere will not have uniform density, making the result of sampling potentially higher resolution near the poles, and lower resolution at the equator. Empirically, we found little difference in training performance between grid sampling and random sampling on the unit sphere in the loss. However, for more detailed objects or reconstruction tasks, this may be an important aspect to examine.

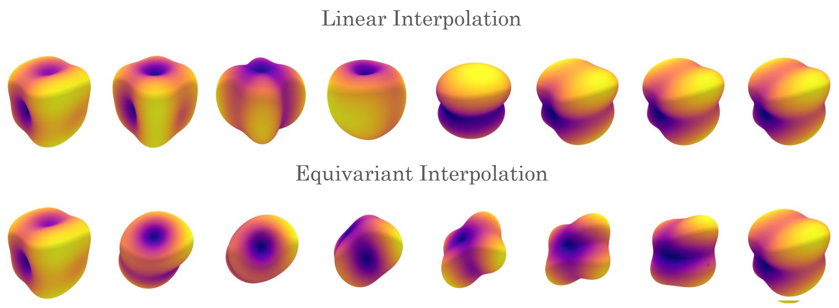

Figure 8: **Interpolation with small latent space and 100 boxes** – We used $l_{max,latent} = 1$ and trained on $n = 100$ shapes. Equivariant interpolation keeps the shape of a box while interpolating while linear interpolation changes the shape.

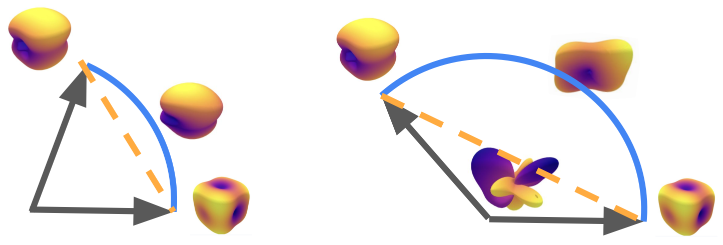

Figure 9: **Linear vs Equivariant Interpolation** – Intuitively, linear interpolation traverses the line but equivariant interpolation traverses the arc. With larger rotations, the difference between these two traversals becomes more significant.

6.3 Training Dynamics

We observe two main issues during training, vanishing gradients and local minimum.

In order to train, we make extensive use of batch normalization in the network architecture. Without this normalization, vanishing gradients were observed after a few epochs. However, the use of normalization potentially reduces the expressiveness of our latent space as both the norm and direction of the latent vectors can encode information. In order to capture this more expressive latent space, we would like to find a way to produce stable training without the use of batch normalization.

We found that training could be split into two main phases: Random to cylinder and cylinder to boxes. Within a few epochs, we found that the network would converge to producing cylinders scaled roughly to the size of the input boxes. This convergence is very robust to parameters such as learning rate, loss function, or even network architecture. However, if training is continued, the network will eventually learn to predict boxes instead of just cylinders. This takes significantly longer, often in the range of 5000 epochs.

We believe that a likely cause of this is that the sharp corners of boxes require the use of higher degree spherical harmonic coefficients, requiring the network to learn to up-sample the latent space more. Future work may evaluate this claim more quantitatively.

6.4 Alternative Architectures

6.4.1 Clebsch-Gordon Networks

In testing, we built a network using tensor products and the Clebsch-Gordon transform. This network was significantly more difficult to train than the Spherical-CNN derived network. We noticed that the forward pass of the network was significantly slower than the Spherical-CNN architecture, likely due to the computational cost of the tensor product and reduce operations.

We found that the Clebsch-Gordon architecture reached the same local minimum as the Spherical-CNN architecture of producing a cylinder of the same size of the box. However, we could not train out of this minimum in any reasonable time.

6.4.2 Tensor Field Networks

A potential issue with the Spherical-CNN approach is that conversion from mesh to coefficients of the spherical harmonics is lossy for any finite l_{max}. In effect, the resolution of the resulting coefficients is only dependent on l_{max} and independent of the input data resolution, making high detail shape generation difficult.

Future work may explore the use of point based representations of objects and architectures such as Tensor Field Networks [17]. This would enable network output resolution to scale with input resolution while preserving the equivariance properties of Spherical-CNNs. Previous works have shown point cloud shape generation without equivariant networks [22], signaling that this should be feasible in the equivariant context as well.

7 Conclusion

Leveraging the SO(3) symmetry in 3D data, we present a preliminary method to increase the data efficiency of 3D shape generation using an SO(3) equivariant autoencoder. We demonstrate that by encoding objects as the coefficients of the spherical harmonics and by devising an equivariant interpolation strategy, we can generate novel shapes when training on objects from our boxes dataset. We also demonstrate robust reconstruction results and describe the relationship between latent dimensionality, training dynamics, and performance in an SO(3) equivariant latent space. Finally, we highlight directions for future work including new interpolation methods and resolution preserving SO(3) equivariant architectures. We hope that this work may serve as a foundation for future works in SO(3) equivariant novel shape generation.

8 Implementation

Code can be found at:
`https://github.com/Shayan-P/so3-equivariance-latent-space-exploration`.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, *et al.*, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[2] P. Henzler, N. J. Mitra, and T. Ritschel, "Escaping plato's cave: 3d shape from adversarial rendering," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 9984–9993.

[3] J. Ho, A. Jain, and P. Abbeel, "Denoising diffusion probabilistic models," *Advances in neural information processing systems*, vol. 33, pp. 6840–6851, 2020.

[4] B. Mildenhall, P. P. Srinivasan, M. Tancik, J. T. Barron, R. Ramamoorthi, and R. Ng, "Nerf: Representing scenes as neural radiance fields for view synthesis," *Communications of the ACM*, vol. 65, no. 1, pp. 99–106, 2021.

[5] K. Schwarz, A. Sauer, M. Niemeyer, Y. Liao, and A. Geiger, "Voxgraf: Fast 3d-aware image synthesis with sparse voxel grids," *Advances in Neural Information Processing Systems*, vol. 35, pp. 33 999–34 011, 2022.

[6] E. R. Chan, M. Monteiro, P. Kellnhofer, J. Wu, and G. Wetzstein, "Pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 5799–5809.

[7] V. Sitzmann, J. Martel, A. Bergman, D. Lindell, and G. Wetzstein, "Implicit neural representations with periodic activation functions," *Advances in neural information processing systems*, vol. 33, pp. 7462–7473, 2020.

[8] A. Yu, S. Fridovich-Keil, M. Tancik, Q. Chen, B. Recht, and A. Kanazawa, "Plenoxels: Radiance fields without neural networks," *arXiv preprint arXiv:2112.05131*, vol. 2, no. 3, p. 6, 2021.

[9] B. Kerbl, G. Kopanas, T. Leimkühler, and G. Drettakis, "3d gaussian splatting for real-time radiance field rendering," *ACM Transactions on Graphics*, vol. 42, no. 4, 2023. [Online]. Available: `https://repo-sam.inria.fr/fungraph/3d-gaussian-splatting/`.

[10] B. Poole, A. Jain, J. T. Barron, and B. Mildenhall, "Dreamfusion: Text-to-3d using 2d diffusion," *arXiv preprint arXiv:2209.14988*, 2022.

[11] A. Karnewar, A. Vedaldi, D. Novotny, and N. J. Mitra, "Holodiffusion: Training a 3d diffusion model using 2d images," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 18 423–18 433.

[12] Y. Shi, P. Wang, J. Ye, M. Long, K. Li, and X. Yang, "Mvdream: Multi-view diffusion for 3d generation," *arXiv preprint arXiv:2308.16512*, 2023.

[13] S. Luo and W. Hu, "Diffusion probabilistic models for 3d point cloud generation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 2837–2845.

[14] T. S. Cohen, M. Geiger, J. Köhler, and M. Welling, "Spherical cnns," *arXiv preprint arXiv:1801.10130*, 2018.

[15] C. Esteves, C. Allen-Blanchette, A. Makadia, and K. Daniilidis, "Learning so (3) equivariant representations with spherical cnns," in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018, pp. 52–68.

[16] R. Kondor, Z. Lin, and S. Trivedi, "Clebsch–gordan nets: A fully fourier space spherical convolutional neural network," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[17] N. Thomas, T. Smidt, S. Kearnes, *et al.*, "Tensor field networks: Rotation-and translation-equivariant neural networks for 3d point clouds," *arXiv preprint arXiv:1802.08219*, 2018.

[18] M. Weiler, M. Geiger, M. Welling, W. Boomsma, and T. S. Cohen, "3d steerable cnns: Learning rotationally equivariant features in volumetric data," *Advances in Neural Information Processing Systems*, vol. 31, 2018.

[19] F. Fuchs, D. Worrall, V. Fischer, and M. Welling, "Se (3)-transformers: 3d roto-translation equivariant attention networks," *Advances in neural information processing systems*, vol. 33, pp. 1970–1981, 2020.

[20] C. Deng, O. Litany, Y. Duan, A. Poulenard, A. Tagliasacchi, and L. J. Guibas, "Vector neurons: A general framework for so (3)-equivariant networks," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 12 200–12 209.

[21] A. D. S. Costa, I. Mitnikov, M. Geiger, M. p Ponnapati, T. Smidt, and J. JACOBSON, *Ophiuchus: Scalable modeling of protein structures through hierarchical coarse-graining SO(3)-equivariant autoencoders*, 2024. [Online]. Available: `https://openreview.net/forum?id=fNOewRJLgQ`.

[22] Y. Zhang, H. Wang, G. Lin, V. C. H. Nicholas, Z. Shen, and C. Miao, "Starnet: Style-aware 3d point cloud generation," *arXiv preprint arXiv:2303.15805*, 2023.