Interpolated SelectionConv for Spherical Images and Surfaces

David Hart, Michael Whitney, Bryan Morse
Brigham Young University
{davidmhart, mikeswhitney, morse}@byu.edu

Abstract

We present a new and general framework for convolutional neural network operations on spherical (or omnidirectional) images. Our approach represents the surface as a graph of connected points that doesn't rely on a particular sampling strategy. Additionally, by using an interpolated version of SelectionConv, we can operate on the sphere while using existing 2D CNNs and their weights. Since our method leverages existing graph implementations, it is also fast and can be fine-tuned efficiently. Our method is also general enough to be applied to any surface type, even those that are topologically non-simple. We demonstrate the effectiveness of our technique on the tasks of style transfer and segmentation for spheres as well as stylization for 3D meshes. We provide a thorough ablation study of the performance of various spherical sampling strategies.

1. Introduction

Omnidirectional (or spherical) images are ever present in modern computer vision, from 360° videos for VR to HDRI images as lighting references in 3D graphics. Much research has gone into trying to bring the benefits of deep learning to this useful domain. A naive approach is to simply apply 2D Convolutional Neural Networks (CNNs) to a projection of the sphere, such as on the equirectangular image or a cubemap unfolding. However, all planar projections of a sphere lead to distortion of the 3D content. Thus, traditional 2D CNNs perform poorly in such cases.

To resolve this, two approaches have been well explored previously. The first is to design the convolution operator uniquely for the sphere, looking at important properties such as rotation equivariance of the convolution operation [3, 4, 14]. The second is to try to transfer what has been learned from a 2D CNN to a specialized network that can operate on spherical images, either through distorting the convolution operator itself [5, 28] or through special projections of the spherical image into the plane [8, 33]. Both of these approaches have potential shortcomings. In the first, by redefining a convolution specific to the spherical domain, all training data must also be in the spherical domain, for which available datasets tend to be much smaller than their 2D counterparts. In the second approach, techniques are often restricted to particular samplings of the sphere (most often the icosphere [8, 33]), which can only exist at specific resolutions, and transitions to these representations may require fine-tuning to properly adjust the CNN weights [33].

Figure 1. Interpolated SelectionConv can perform tasks such as segmentation on spheres (left) and 3D mesh stylization (right).

This work aims to begin to bridge the gap between these two approaches to operating on omnidirectional images. We propose a framework that builds on our previous work of SelectionConv [13], a graph-based approach for transferring weights from a 2D CNN. By using an edge-interpolated version of this method, we can create a graph of points on the sphere using any sampling scheme that we desire and still maintain the anisotropic behavior of convolution that is needed for transferring from 2D CNNs. Our new approach is general while remaining fast and efficient since it builds on previous graph network infrastructure.

We show that our method is a simple and effective way to bring 2D techniques into the spherical domain, while avoiding many of the drawbacks of regular SelectionConv and previous approaches. We also show that customizable sampling techniques may be more effective than the icosphere approach that has generally been used. Lastly, while we primarily focus on spherical images for this work, our method is general enough to operate on any surface or 3D mesh. Examples of tasks that can be performed with our method are shown in Fig. 1.

In summary, our contributions are as follows:

- We introduce an interpolation-based scheme for using SelectionConv in non grid-like spaces.

- We adapt Interpolated SelectionConv to work for surface point-cloud representations by modifying the selection function and clustering method.
- We demonstrate the effectiveness of our graph representation with customizable sampling techniques on the tasks of spherical stylization and segmentation. We also demonstrate stylization of 3D meshes. We achieve results comparable to state-of-the-art methods.
- We present a thorough ablation study of various spherical sampling and clustering techniques.

2. Related Works

2.1. SelectionConv

This work builds heavily on the idea of selection-based convolution, which was first proposed in SelectionConv [13], a method for using CNNs on irregular image types such as cubemaps and masked images. In selection-based convolution, graph networks can be manipulated to be commensurate with 2D CNNs. This is done by preprocessing a graph's adjacency matrix into multiple adjacency matrices, where each matrix represents a cardinal or ordinal direction. By doing this, spatial structure is built back into the graph and weights can be transferred from a trained 2D CNN to a modified graph network. Thus, graph structures can be designed for specific irregular image types and the same 2D networks can operate in these new domains.

In [13], all experiments were done with graph representations that assume a single selection per target node. This allowed graph connections to go across seams for irregular images, but a single selection is not sufficient when building the graph in more general spaces such as arbitrary surfaces or point clouds. We extend this approach by demonstrating how to effectively allow multiple selections per node to give interpolated values when selections lie between discrete spatial locations. This is similar to techniques such as [20], but with the added benefit that we can orient the interpolation for each individual node on a surface.

2.2. Spherical CNNs

Neural networks designed for spheres fall roughly into two categories. The first is projection-based CNNs that aim to learn on traditional 2D images and then utilize that information in some way on the sphere. The second is surface-based CNNs that have special convolution operations designed specifically for spheres.

Projection-based CNNs

The first approaches to applying CNNs to the sphere operated on projections of the sphere such as the equirectangular projection [17], cubemap projection [2, 21], or even a combination of multiple projections [15, 25, 29]. After the proposal of deformable convolution by Dai *et al*. [6], many researchers focused on deforming the convolution operator specifically for sphere projections [5, 26, 28].

Lee *et al*. [18] later demonstrated the effectiveness of operating on an icosahedral approximation of the sphere. Zhang *et al*. [33] adapted a similar representation, but allow for a more direct CNN operation by breaking the icosahedron into 5 planar projections. Eder *et al*. [8] increased performance on many spherical tasks by representing the sphere as a set of tangent images at the icosphere vertices.

Surface-based CNNs

Cohen *et al*. [3, 4] have proposed a convolution structure designed specifically for spheres and surfaces that is rotation and gauge equivariant. Representations have also been designed that utilize spherical harmonics [9] or define the convolution in terms of a linear combination of differential operators [14]. This line of work continues to expand to be used for more complex and general spherical networks, such as LSTMs [31].

Our method aligns closely with projection-based approaches since it transfers weights from a 2D CNN and operates with oriented convolutions on the sphere. Our approach, however, builds on a framework that is not limited to spheres, and, like surface-based CNNs, is not constrained to a specific sampling technique.

2.3. Sphere and Mesh Style Transfer

Many spherical approaches demonstrate performance on tasks such object detection, semantic segmentation, and depth prediction. While we compare our method on the task of segmentation, we also demonstrate that our representation generates high-quality stylizations of the sphere. Ruder *et al*. [23] were the first to demonstrate style transfer on a spherical image, but their method requires a slow optimization approach or fine-tuning a style network for spheres. Like [13], our approach can stylize the sphere in a single feed-forward pass with any content and style image, but we do so with fewer artifacts than previous approaches.

Since our representation can easily be applied to general surfaces, we demonstrate style transfer for meshes as well. There have been many approaches to applying convolutions to surfaces and 3D meshes [11, 12, 16, 24, 30], but these convolutions are trained on 3D data for 3D-specific tasks. Some methods have performed mesh style transfer [10, 13, 27, 32], but these approaches are optimization-based, require professional reference renderings, use aligned texture maps, or have artifacts at seams. Our approach is simple, fast, and has no model or style prerequisites.

3. Interpolated SelectionConv

As discussed in Sec. 2.1, our method builds on the original SelectionConv [13], which proposes a modified graph network that can distinguish between the edges of a graph

during a convolution operation. Thus, if a graph is designed and processed for a specific type of irregular image, selection-based graph convolution will preserve the orientation and allow a traditional CNN to operate even in the irregular space.

In order to distinguish the edges during a graph convolution, the graph must be preprocessed into multiple adjacency matrices. Selection-based convolution extends the common formulation of graph convolution to reflect these different spatially-relative adjacency matrices. Specifically,

$$\mathbf{X}^{(k+1)} = \sum_m \tilde{\mathbf{S}}_m \mathbf{X}^{(k)} \mathbf{W}_m \tag{1}$$

where m represents a given direction/selection, $\tilde{\mathbf{S}}_m$ is the adjacency matrix for that selection, $\mathbf{X}^{(k)}$ is the current node activations, and $\mathbf{W}_m$ is the learned weights for that selection. Rather than a binary adjacency matrix, note that $\tilde{\mathbf{S}}_m$ is normalized so that multiple incoming edges can have the same selection, allowing for more general point-cloud-like structures. This work further utilizes this normalization to break the restriction that each edge have a single selection.

In our original SelectionConv work, edges were given the selection of the cardinal or ordinal direction most closely aligned with the spatial relationship between the respective pixels or nodes. With general sampling patterns, however, node relationships do not lie so cleanly along specific axes. Thus, it is beneficial to be able to interpolate between multiple selections. To do this, we first assign multiple edges to a single node with each edge having a different selection (i.e., we assign the same source and target node to multiple adjacency matrices). Then, we assign each of those edges an edge weight that determines how much the spatial relationship between the nodes matches the given direction.

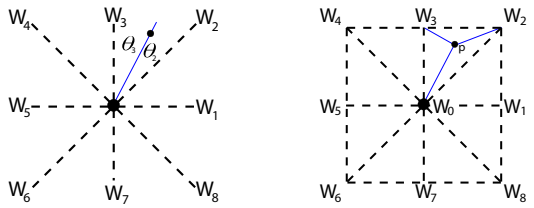
With this, we have two different senses of normalization. The first accounts for our introduced interpolation by normalizing all the assigned edges for a given source and target node, making the total of the interpolation weights across selections equal to one, or mathematically:

$$\hat{S}_m(i,j) = \frac{S_m(i,j)}{\sum_k S_k(i,j)} \tag{2}$$

This guarantees that a single target node can not have an unbalanced amount of influence during the aggregation step. The second normalizes in the manner discussed in [13], where each of the adjacency matrices is normalized individually so that for a given source node, the sum of all the target node edge weights equals one, or mathematically:

$$\tilde{S}_m(i,j) = \frac{\hat{S}_m(i,j)}{\sum_k \hat{S}_m(i,k)} \tag{3}$$

With this formulation, we are no longer constrained to discrete selections but can interpolate selection values within the convolution operation. There are various ways to interpolate a location's value given a set of points. We demonstrate two simple and effective ways to interpolate selection weights given source and target nodes.



a) Angular Interpolation b) Barycentric Interpolation

Figure 2. Illustration of a) angular interpolation versus b) barycentric interpolation. In angular interpolation, the edge to each node is added to the adjacency matrix of the two selections it lies between. If a local distance/radius can be determined, barycentric interpolation can be used which also includes the central selection.

3.1. Angle-based Interpolation

One way to practically determine the interpolation of a node between two selections is to use angles. Instead of simply choosing the selection in the direction that most closely matches a given edge, we use the angles between the given edge and the unit vectors for each selection direction (excluding the central selection). The two selections, a and b, with the smallest angles are added to the respective adjacency matrices and the edge weights are equal to the convex combination of the two angles, or mathematically:

$$w_a = \frac{\theta_b}{\theta_a + \theta_b} \tag{4}$$

$$w_b = 1 - w_a \tag{5}$$

This process is illustrated in Fig. 2.a.

3.2. Barycentric Interpolation

The angle-based interpolation is intuitive and would be expected to perform well for graph structures where nodes are spaced approximately equidistant from each other. However, another interpolation scheme can be used that accounts properly for distance from the source node. If an average or expected distance between nodes can be calculated, then the interpolation can be triangulated between three selections using barycentric interpolation. This process is illustrated in Fig. 2.b.

Though the general form of barycentric interpolation requires inverse matrices or other time-consuming calculations, it can be dramatically simplified since the three points of interpolation will always lie along a perfect right triangle with two bases of length d. In these circumstances, the three edge weights become

$$w_0 = 1 - \frac{\max(|p|)}{d} \tag{6}$$

$$w_a = \frac{\max(|p|) - \min(|p|)}{d} \tag{7}$$

$$w_b = \frac{\min(|p|)}{d} \tag{8}$$

where p is the location of the target node relative to the source node. Further details and a proof of this formulation are shown in Appendix A of the supplemental material.

4. The Spherical Graph

As noted in Sec. 2.2, many networks that operate on spherical images must first project the spherical data to some form. Ideally, rather than working on projections, it would be best to work on the spherical data directly without relying on projection. Thus, we generate our graph structure in 3D space using the spherical surface and adjust our selection function appropriately. We will now give the details of how we generate the exact graph structure.

4.1. The Spherical Selection Function

The original selection function used in [13] is

$$s(v_i, v_j) = \begin{cases} 0 \text{ if } \|\mathbf{x}_j - \mathbf{x}_i\| < \epsilon \\ \underset{k}{\operatorname{argmax}}\ \mathbf{D}_k \cdot (\mathbf{x}_j - \mathbf{x}_i) \text{ otherwise} \end{cases} \tag{9}$$

where v_i and v_j are two nodes of the graph, $\mathbf{x}_i \in \mathbb{R}^2$ and $\mathbf{x}_j \in \mathbb{R}^2$ are their respective positions, and $\{\mathbf{D}_k \in \mathbb{R}^2, 1 \leq k \leq 8\}$ represents the set of unit vectors for each of the 8 cardinal/ordinal directions. In order to work on spheres, we modify this selection function by making selections relative to the surface of the sphere, using a local planar approximation to determine orientation.

To make a local planar approximation, we first need the normal $\hat{\mathbf{z}}$ to each node position (which is conveniently the normalized version of the 3D location $\mathbf{x}_i \in \mathbb{R}^3$ for a sphere). Next, we generate a rotation matrix using the approximate up-vector of $\tilde{\mathbf{y}} = \langle 0, 1, 0 \rangle$ and Graham-Schmidt orthogonalization. Specifically,

$$\hat{\mathbf{x}} = \tilde{\mathbf{y}} \times \hat{\mathbf{z}} \tag{10}$$

$$\hat{\mathbf{y}} = \hat{\mathbf{z}} \times \hat{\mathbf{x}} \tag{11}$$

$$\mathbf{R} = \begin{bmatrix} \hat{\mathbf{x}} \\ \hat{\mathbf{y}} \\ \hat{\mathbf{z}} \end{bmatrix} \tag{12}$$

With this rotation matrix, edges from the node can be rotated relative to the surface. Then, the new z-coordinate is dropped, giving a local planar approximation of the coordinate relative to the surface. This dropping of z-coordinate can be represented by multiplying by a truncated identity matrix:

$$\mathbf{I}_t = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \tag{13}$$

a) Spherical Graph b) Local Planar Projection

Figure 3. Visualization of the graph connections being made in 3D space (a), and selections being made based on the local planar projection at each node (b).

Thus, the final spherical selection function (for a single selection) becomes

$$s(v_i, v_j) = \begin{cases} 0 \text{ if } \|\mathbf{x}_j - \mathbf{x}_i\| < \epsilon \\ \underset{k}{\operatorname{argmax}}\ \mathbf{D}_k \cdot \mathbf{I}_t \mathbf{R} (\mathbf{x}_j - \mathbf{x}_i) \text{ otherwise} \end{cases} \tag{14}$$

Once the main selection is made, other selections and interpolation values can be determined according to Sec. 3. An illustration of our final graph structure is shown in Fig. 3.

4.2. Sampling

Most spherical data is stored in the form of an equirectangular image, where the spherical data has been projected to a cylinder and then unfolded to the plane. This representation is easy to store, but the pixels do not represent locations that are equidistantly sampled across the sphere. Preferably, the algorithm should sample points from the image in such a way that their 3D locations are more consistently spaced across the surface.

For this work, we consider the following sampling strategies when comparing to the equirectangular baseline:

- **Fibonacci Spiral**: A common and fast approach for approximating equidistant points.
- **Icosphere**: Points generated by taking the icosahedron and subdividing iteratively.
- **Layering**: A sampling method described in [7], where points are appropriately distanced in θ based on equidistant ϕ values.

Examples of each of these sampling schemes are shown in Fig. 4. We include the layering sampling algorithm here (Algorithm 1) for convenience, which we adapt from [7].

4.3. Clustering

Nodes in the graph need to be clustered together for the downsampling stages of the CNNs. When operating on irregular images as in [13], these could be easily determined by grouping pixels together on a 2D grid. However, determining clusters for a sphere based on the 2D equirectangular image removes the benefits of working in a 3D space.

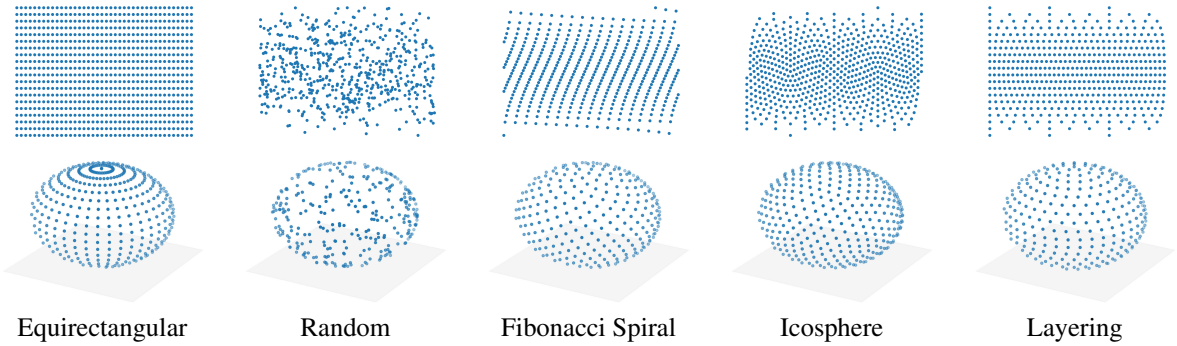


Figure 4. The various sampling methods we test for building our graph on the spherical surface. The sampling points on the equirectangular image (top) correspond to particular points on the spherical surface (bottom).

Algorithm 1 Layering Sampling

Require: N_ϕ, (i.e. number of sampling rows)
$a \leftarrow \pi^2 / N_\phi^2$
$d \leftarrow \sqrt{a}$
$M_\phi \leftarrow \text{round}[\pi / d]$
$d_\phi \leftarrow \pi / M_\phi$
$d_\theta \leftarrow a / d_\phi$
for each m in $0...M_\phi - 1$ **do**
  $\phi \leftarrow \pi(m + 0.5)/M_\phi$
  $M_\theta \leftarrow \text{round}[2\pi \sin(\phi / d_\theta)]$
  for each n in $0...M_\theta - 1$ **do**
    $\theta = 2\pi n / M_\theta$
    Append point (θ, ϕ)
  end for
end for

Thus, to cluster in 3D, we must resample the sphere at a lower resolution. Each current node is clustered to the resampled node of closest proximity.

All of the previous sampling techniques (as well as random sampling) can be used, with an aim for a desired cluster size. For example, a stride of 2 in each dimension is commonly used in downsampling layers of CNNs. This would give a desired cluster size of 4, and each of the sampling methods would be used in the following ways:

- **Random**: Randomly generate 1/4 as many points.
- **Fibonacci Spiral**: Resample with 1/4 as many points.
- **Icosphere**: Complete one less subdivision of the icosahedron (since 1/4 as many points will be generated).
- **Layering**: Increase the distance between ϕ values by 2 and resample.

4.4. Customizable Resolution

When 2D CNNs are trained, there is generally a fixed resolution that the input data is set too. Additionally, there is usually a small range of field-of-views in the cameras that were used to generate the training data. In a 2D sense, this intrinsically sets an expected size and scale that the network is expecting to see certain objects. While data augmentation and other techniques can make the network more flexible to some degree, it is important that our graph structures lie at relatively the same resolution and field-of-view that the network is expecting.

For example, the Stanford2D-3D-S dataset [1] that we evaluate on in Sec. 6 uses training images that have a field-of-view (FOV) between 45° and 75°. If a 2D CNN was trained on N × N images of that camera, the relative angle between points on our 3D surface in any direction should be approximately:

$$\Delta\theta = \frac{\text{FOV}}{N} \tag{15}$$

This spacing is handled by the sampling method we use for the sphere. As mentioned previously, most methods have relied on icosphere sampling for their approaches, but the icosphere is constructed through subdivisions and thus is limited to discrete resolutions that are quadruplings of the initial resolution. With a sampling approach such as layering, however, we can directly control the sampling density on the sphere and thus match the desired $\Delta\theta$ more precisely. This gives it improved performance over the icosphere, which we demonstrate in Sec. 7.

5. Surface Graphs

The approach described in Sec. 4 was specific to spheres, but with a few simple modifications, it can be applied to any general surface, specifically 3D meshes. To generate the graph on a surface, we start by randomly sampling the surface. For each node, the normal of the face it was sampled from and its respective UV coordinate are also stored. Next, edges connecting nearby points are made using a KNN approach (while also accounting for folds by culling points with disparate normals).

Then, to determine selections, the same selection process and selection function as Sec. 4.1 can be used with two key differences. First, the approximate up-vector for the

surface is either manually assigned (if a reasonable volume orientation exists) or is randomly generated. Second, the normal $\hat{\mathbf{z}}$ used for the Graham-Schmidt orthogonalization is the stored face normal for each node. All other steps in the process remain the same for the graph generation. The surface is then resampled at a lower resolution to determine clustering nodes, and the whole process is repeated for the needed number of downsamplings.

As is shown in Sec. 6.3, one benefit of this approach is that, just like the sphere, we can customize the approximate resolution of the point cloud on the mesh by controlling the number of points in the initial sampling. Increasing the number of initial points increases the level of detail to which we operate on the mesh.

6. Results

For the results presented, we use the layering method for both sampling and clustering discussed in Sec. 4 and the angular interpolation presented in Sec. 3.1. We also use replicate padding in the networks to handle missing selections. We use this representation for its ability to customize the resolution on the sphere and since we found it to perform the best overall on all tasks. A more thorough analysis of the different sampling and clustering techniques, as well as a comparison between the interpolation techniques, is presented in Sec. 7.

6.1. Spherical Stylization

We start with a clear visual example of the effectiveness of transferring weights from a 2D CNN to our graph structures without the need of any additional fine-tunning. We demonstrate style transfer with the approach proposed by Li *et al*. [19] on our spherical representations. SelectionConv [13] achieved spherical style transfer by building a fully connected graph structure based on a cube map. While this approach improved the stylization over the naive method of stylizing the equirectangular image directly, the uninterpolated nature of the graph can give artifacts along pixels where a direction transition occurs. Our new approach removes such artifacts, as shown in Fig. 5. This is possible because we can smoothly vary the selection between nodes in 3D space. Additionally, when reconstructing the equirectangular image, the relevant points are interpolated in 3D space, which gives a smooth and plausible stylization throughout the whole sphere. Our new method is also similar to [13] in run time and memory usage.

6.2. Spherical Segmentation

A common spherical image task is omnidirectional semantic segmentation. We evaluate our method on such a task using the Stanford 2D-3D-S dataset [1]. It consists of equirectangular images of various indoor scenes with 13 different classes.

Method	Input	mIOU	fine-tuned
Cubemap [13]	RGB	36.3%	-
UGSCNN [14]	RGB-D	-	38.3%
GaugeNet [3]	RGB-D	-	39.4%
Sphere (Ours)	RGB-D	39.9%	**41.4%**
HexRUNet [33]	RGB-D	-	43.3%
TangentIms [8]	RGB-D	38.9%	51.9%

Table 1. Mean intersection-over-union (mIOU) on the Stanford 2D-3DS dataset segmentation task [1]. The mIOU is shown for both direct transfer (left) and after fine-tuning (right). Our method performs comparably to state-of-the-art approaches.

We first train a simple 6-layer deep U-Net architecture [22], similar to the architecture used in [14, 33], and include depth information with the input to be consistent with state-of-the-art methods. We train on the provided precomputed views of the spherical image. Once the network has been trained in 2D space, we transfer over the weights to our graph representation with interpolated selections. We use the layering approach and scale the resolution to match the average $\Delta\theta$ from the training data.

We can also fine-tune the graph representation on the full 3D data. These fine-tuning steps can be completed quickly since our graph structure only needs to be calculated once, then it can be reused in every successive iteration. Only the node feature values need to be updated.

The mean IOU for our method, both before and after fine-tuning in spherical space, is shown in Table 1, along with comparisons to other previous works. We evaluate on the full scale equirectangular images (4096×2048) and follow the standard practice of weighting the metrics for each pixel by cosine of their latitude location, similar to [14, 33].

As is shown, our method has performance comparable to state-of-the-art approaches, even after a direct transfer without fine-tuning. While [33] and [8] have higher mean IOU scores, we note that [33] must use a fine-tuned icosphere representation and that [8] requires processing 80 different 512×512 planar projections to make a single spherical image, whereas our approach can evaluate in a single pass with a reusable graph structure between spherical images.

6.3. Mesh Stylization

Lastly, we demonstrate the ability of our method to generalize to general surfaces by stylizing 3D meshes. In [13], a UV edge-pairing process was required to determine edges across seams in the texture map. This process failed to remove all artifacts along seam boundaries and was time consuming for high-polygon-count meshes. Our new approach avoids this computationally expensive step while achieving far better consistency along UV seams since the edges and selections are made in 3D space, rather than on the topologically complex 2D texture map. A comparison of this approach to the previous work is shown in Fig. 6. Addi-

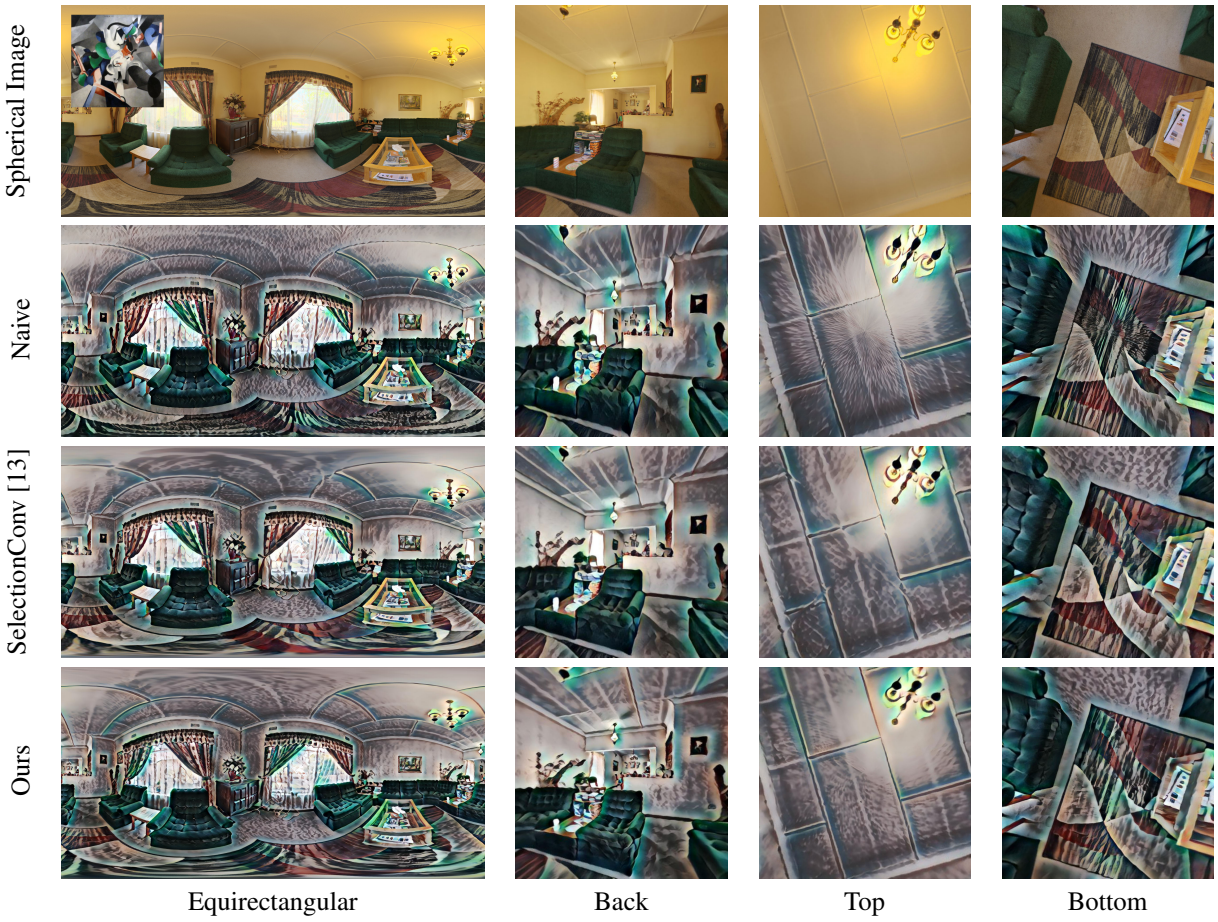
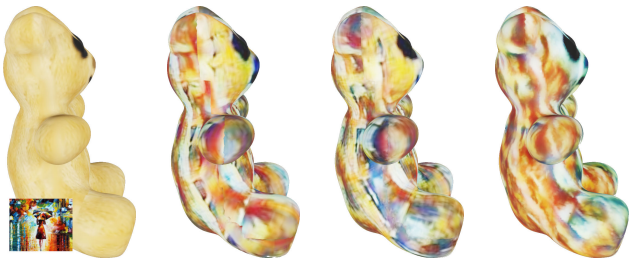


Figure 5. A 360° image (1st row), its stylization when naively stylizing the equirectangular image (2nd row), using the cube-map graph setup in [13] (3rd row), and compared to our interpolated spherical representation (4th row). The equirectangular projection along with various views of the scene are presented. In the naive approach, note the vertical seam in the middle of the back view as well as the distortion in the top and bottom views. In the original SelectionConv results, note the artifacts in the top and bottom views along the seam connections (making an x shape). Those artifacts are removed with our new method. Public domain image courtesy of polyhaven.com.



a) Mesh b) Naive c) [13] d) Ours

Figure 6. When the texture map of the original mesh (a) is stylized naively (b), many artifacts are present along the UV seams. Stylizing with SelectionConv (c) removed some of those artifacts, but inconsistencies remain. Interpolated SelectionConv (d) retains far greater consistency along seams.

tionally, our method allows users to control the number of initial sampling points, making it time invariant to the complexity of the mesh. This also provides a simple way for controlling the level of detail in the stylization on the mesh while operating on high quality meshes and texture maps. Examples of these benefits are shown in Fig. 7 and in the supplemental material.

7. Ablation Study

We presented the layering sampling and clustering technique (with angular interpolation) as our best representation of the sphere. However, one of the benefits of our method is that we are not constrained to any specific graph structure. Thus, we give the performance of the other possible representations of our sphere.

We test the various sampling methods and clustering methods with the previously presented segmentation task

Sampling\Clustering	Random	Equirectangular	Spiral	Icosphere	Layering
Equirectangular	35.1% / 35.0%	38.1% / 37.0%	38.9% / 38.8%	38.7% / 38.5%	40.7% / 39.8%
Spiral	32.8% / 32.5%	32.8% / 32.3%	35.7% / 35.9%	36.3% / 36.2%	37.9% / 37.2%
Icosphere	31.9% / 30.8%	33.3% / 31.9%	35.9% / 35.4%	36.5% / 35.6%	38.1% / 36.7%
Layering	34.3% / 33.6%	36.2% / 34.3%	38.1% / 37.5%	38.8% / 38.2%	**39.9%** / 38.7%

Table 2. Ablation study comparing mIOU results for the various sampling, clustering, and interpolation approaches on the Stanford2D-3D-S segmentation task. Angle-based interpolation results are shown on the left. Barycentric results are shown on the right. Results for random clustering were averaged over multiple configurations.

a) Mesh b) Naive c) Ours

Figure 7. Example stylizations of high quality meshes with 4K textures (a). Naively stylizing the texture map (b) is slow, leaves artifacts on seams, and provides little control for the level of detail for the stylization. Our approach (c) removes seam artifacts while allowing the user to control the number of sampling points, which in turn determines the speed and detail of the stylization. Public domain meshes courtesy of polyhaven.com.

without fine-tuning. Additionally, we compare barycentric interpolation versus angle-based interpolation for these various representations. The results are shown in Table 2.

The results show some interesting patterns. First, all methods perform worst when using random and equirectangular clustering. This is to be expected since the points will not be pooled together based on equidistant locations. Second, we note that spiral and icosphere sampling perform poorest overall since the spiral approach generates neighborhoods that tend to be more irregular and neither allows for customizing the resolution in a precise way. (We also note that icosphere performs more poorly than demonstrated elsewhere, but we attribute this to the fact that we do not explicitly modify the convolution operator to be hexagonal like many icosphere-specific methods [18, 33].) Also, barycentric interpolation performs slightly below angle-based interpolation for most sampling and clustering representations. We attribute this to the blurring effect that comes from the larger number of edges, especially when compared to the angle-based representation when most points are reasonably spaced. Lastly, we note that layering clustering did the best for all methods. Equirectangular sampling with layering clustering even outperformed the layering/layering representation on the segmentation task, but we note that it used 25% more nodes and edges, which makes it slower to run and harder to fine-tune.

8. Conclusion

We have presented a general framework for transferring weights from 2D convolutional networks to graph networks that can operate on spherical images and surfaces. These networks can be fine-tuned even further in their specific domains as needed. This approach allows for a simple and effective way to improve performance on spherical tasks without requiring large datasets that are specific to the spherical domain. We have also demonstrated possible applications for 3D meshes and have provided a thorough ablation study exploring sampling of spherical surfaces.

Limitations

The main limitation of our approach comes from the extra memory needed to store the graph edges while performing the convolution. Though we describe our method mathematically using adjacency matrices, the implementation actually uses memory-efficient edge indexes. However, this means that allowing multiple selections for the same edge increases the size of the edge index. Additionally, since the interpolation values also need to be stored, the memory needed for a graph with many nodes can become quite large. Since our focus was on direct transfer, this is usually not an issue and most graphs can still fit in GPU memory, but fine-tuning any task must be done with a very small batch size to be feasible.

Like the original SelectionConv, we can't use larger convolution kernel sizes because the edge hopping process is not differentiable. Thus the networks we used contained only 3×3 and 1×1 kernels. We note, however, that this limitation is shared with other previous methods [14, 33].

Future Work

Though we demonstrated style transfer on meshes, the field of meshes and general surfaces is extensive and has many applications. More exploration and experiments using our interpolated SelectionConv in this area could be the study of future work. Looking into additional interpolation schemes and techniques for determining expected radial distances is also worth investigation.

References

[1] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*, Feb. 2017.

[2] Hsien-Tzu Cheng, Chun-Hung Chao, Jin-Dong Dong, Hao-Kai Wen, Tyng-Luh Liu, and Min Sun. Cube padding for weakly-supervised saliency prediction in 360° videos. In *Conference on Computer Vision and Pattern Recognition*. IEEE, June 2018.

[3] Taco Cohen, Maurice Weiler, Berkay Kicanaoglu, and Max Welling. Gauge equivariant convolutional networks and the icosahedral CNN. In *International Conference on Machine Learning*, pages 1321–1330. PMLR, 2019.

[4] Taco S. Cohen, Mario Geiger, Jonas Köhler, and Max Welling. Spherical CNNs. In *International Conference on Learning Representations*, 2018.

[5] Benjamin Coors, Alexandru Paul Condurache, and Andreas Geiger. SphereNet: Learning spherical representations for detection and classification in omnidirectional images. In *European Conference on Computer Vision*, September 2018.

[6] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *International Conference on Computer Vision*, pages 764–773. IEEE, 2017.

[7] Markus Deserno. How to generate equidistributed points on the surface of a sphere, 2004.

[8] Marc Eder, Mykhailo Shvets, John Lim, and Jan-Michael Frahm. Tangent images for mitigating spherical distortion. In *Conference on Computer Vision and Pattern Recognition*, pages 12426–12434. IEEE, 2020.

[9] Carlos Esteves, Christine Allen-Blanchette, Ameesh Makadia, and Kostas Daniilidis. Learning SO(3) equivariant representations with spherical CNNs. In *European Conference on Computer Vision*, pages 52–68, 2018.

[10] Jakub Fišer, Ondřej Jamriška, Michal Lukáč, Eli Shechtman, Paul Asente, Jingwan Lu, and Daniel Sýkora. StyLit: Illumination-guided example-based stylization of 3D renderings. *ACM Trans. Graph.*, 35(4):92:1–92:11, 2016.

[11] Niv Haim, Nimrod Segol, Heli Ben-Hamu, Haggai Maron, and Yaron Lipman. Surface networks via general covers. In *International Conference on Computer Vision*. IEEE, October 2019.

[12] Rana Hanocka, Amir Hertz, Noa Fish, Raja Giryes, Shachar Fleishman, and Daniel Cohen-Or. MeshCNN: a network with an edge. *ACM Transactions on Graphics (TOG)*, 38(4):1–12, 2019.

[13] David Hart, Michael Whitney, and Bryan Morse. SelectionConv: convolutional neural networks for non-rectilinear image data. In *European Conference on Computer Vision*, October 2022.

[14] Chiyu "Max" Jiang, Jingwei Huang, Karthik Kashinath, Prabhat, Philip Marcus, and Matthias Nießner. Spherical CNNs on unstructured grids. In *International Conference on Learning Representations (Poster)*, 2019.

[15] Hualie Jiang, Zhe Sheng, Siyu Zhu, Zilong Dong, and Rui Huang. Unifuse: Unidirectional fusion for 360° panorama depth estimation. *Robotics and Automation Letters*, 6(2):1519–1526, 2021.

[16] Alon Lahav and Ayellet Tal. Meshwalker: Deep mesh understanding by random walks. *ACM Transactions on Graphics (TOG)*, 39(6):1–13, 2020.

[17] Wei-Sheng Lai, Yujia Huang, Neel Joshi, Christopher Buehler, Ming-Hsuan Yang, and Sing Bing Kang. Semantic-driven generation of hyperlapse from 360 degree video. *Transactions on Visualization and Computer Graphics*, 24(9):2610–2621, 2018.

[18] Yeonkun Lee, Jaeseok Jeong, Jongseob Yun, Wonjune Cho, and Kuk-Jin Yoon. SpherePHD: Applying CNNs on a spherical PolyHeDron representation of 360° images. In *Conference on Computer Vision and Pattern Recognition*, pages 9173–9181. IEEE, 2019.

[19] Xueting Li, Sifei Liu, Jan Kautz, and Ming-Hsuan Yang. Learning linear transformations for fast arbitrary style transfer. In *Conference on Computer Vision and Pattern Recognition*. IEEE, 2019.

[20] Jiageng Mao, Xiaogang Wang, and Hongsheng Li. Interpolated convolutional networks for 3d point cloud understanding. In *International Conference on Computer Vision*. IEEE, October 2019.

[21] Rafael Monroy, Sebastian Lutz, Tejo Chalasani, and Aljoscha Smolic. SalNet360: Saliency maps for omni-directional images with CNN. *Signal Process. Image Commun.*, 69:26–34, 2018.

[22] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-Net: Convolutional networks for biomedical image segmentation. In Nassir Navab, Joachim Hornegger, William M. Wells, and Alejandro F. Frangi, editors, *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2015*, pages 234–241, Cham, 2015. Springer International Publishing.

[23] Manuel Ruder, Alexey Dosovitskiy, and Thomas Brox. Artistic style transfer for videos and spherical images. *International Journal of Computer Vision*, 126(11):1199–1219, 2018.

[24] Ayan Sinha, Jing Bai, and Karthik Ramani. Deep learning 3D shape surfaces using geometry images. In *European Conference on Computer Vision*, pages 223–240. Springer, 2016.

[25] Yu-Chuan Su and Kristen Grauman. Learning spherical convolution for fast features from 360° imagery. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, page 529–539, Red Hook, NY, USA, 2017. Curran Associates Inc.

[26] Yu-Chuan Su and Kristen Grauman. Kernel transformer networks for compact spherical convolution. In *Conference on Computer Vision and Pattern Recognition*, pages 9442–9451. IEEE, 2019.

[27] Daniel Sýkora, Ondřej Jamriška, Ondřej Texler, Jakub Fišer, Michal Lukáč, Jingwan Lu, and Eli Shechtman. StyleBlit: Fast example-based stylization with local guidance. *Computer Graphics Forum*, 38(2):83–91, 2019.

[28] Keisuke Tateno, Nassir Navab, and Federico Tombari. Distortion-aware convolutional filters for dense prediction in panoramic images. In *European Conference on Computer Vision*, September 2018.

[29] Fu-En Wang, Yu-Hsuan Yeh, Min Sun, Wei-Chen Chiu, and Yi-Hsuan Tsai. Bifuse: Monocular 360 depth estimation via bi-projection fusion. In *Conference on Computer Vision and Pattern Recognition*. IEEE, June 2020.

[30] Ruben Wiersma, Elmar Eisemann, and Klaus Hildebrandt. CNNs on surfaces using rotation-equivariant features. *ACM Transactions on Graphics (ToG)*, 39(4):92–1, 2020.

[31] Yanyu Xu, Ziheng Zhang, and Shenghua Gao. Spherical DNNs and their applications in 360° images and videos. *Transactions on Pattern Analysis and Machine Intelligence*, PP, 2021.

[32] Kangxue Yin, Jun Gao, Maria Shugrina, Sameh Khamis, and Sanja Fidler. 3DStyleNet: Creating 3D shapes with geometric and texture style variations. In *International Conference on Computer Vision*. IEEE, 2021.

[33] Chao Zhang, Stephan Liwicki, William Smith, and Roberto Cipolla. Orientation-aware semantic segmentation on icosahedron spheres. In *International Conference on Computer Vision*. IEEE, October 2019.