Deep learning 3D shape surfaces using geometry images - Supplementary material

Ayan Sinha[†], Jing Bai[*] and Karthik Ramani[†]

Purdue University[†], Beifang University of Nationalities[*]
{sinha12,bai58,ramani}@purdue.edu

This document serves a supplementary material to the paper, Learning 3D shape surfaces using geometry images. We discuss additional details for creating a geometry image with focus on mesh processing and provide justification of technical parameters for learning on the ModelNet10 dataset. We also provide a MATLAB implementation for creating a geometry image from a spherical parametrization.

1 Mesh processing for creating a geometry image

As our method for authalic parametrization of a shape draws from discrete differential geometry, it is critical that the mesh model is accurate as mentioned in the main manuscript, i.e., it follows the Euler characteristic, given by:

$$2 - 2m = |V| - |E| + |F| \tag{1}$$

Here $|x|$ indicates the cardinality of feature x and m is the genus of the surface. Unfortunately, most mesh models in the Princeton ModelNet benchmark do not strictly follow the Euler characteristic. In such a scenario, we first find *active points* in a grid structure with suitable resolution which are interior to or on the boundary of the mesh model. This is akin to the process of voxelization. We then find the α-shape with radius, r, compatible with the resolution of the voxel grid. An α-shape is a generalization of the convex hull with radius parameter r such that varying r from 0 to ∞ returns a range of shapes between the empty shape and convex hull, respectively. We find the largest connected region of the α-shape with radius r, and use the corresponding boundary vertices and facets for subsequent procedures. The resulting shape is usually accurate barring degeneracies in the original model.

As we do a spherical parametrization of the mesh, it is also critical that the mesh is of genus zero. If the genus of an accurate mesh model is evaluated to be non-zero, we then find the medial axis of the shape and sequentially fill all identified loops until the genus becomes zero. We keep track of the imputed points which constitute the topological mask geometry image of the shape. If the medial axis approach fails to return a genus-zero surface, we progressively increase the radius r of the α-shape till the genus becomes zero. We get a high fidelity representation of the original shape respecting its overall structure after this procedure. Details for creating a topological mask geometry image for non genus zero surfaces which serves as an input feature to the convolutional neural networks is discussed next.

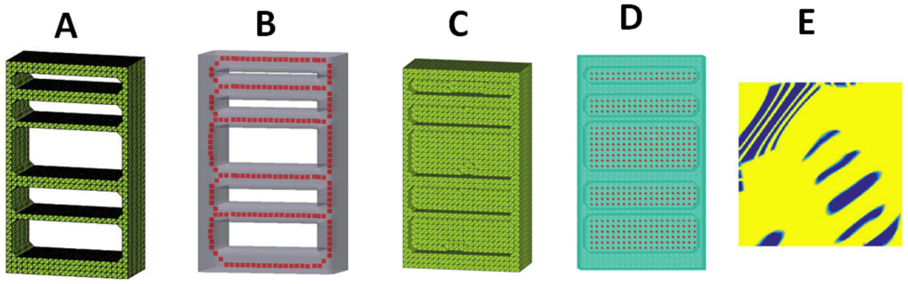

Fig. 1. Medial axis approach to filling topological holes. A: The original shape of bookshelf B: The medial axis of the bookshelf C:The transformed shape by filling the topological holes identified using the medial axis D: The 0-1 index corresponding to newly introduced and original points. The red points are indexed 0 and all other points in the original shape are indexed 1. E: The geometry image corresponding to the 0-1 indexed shape. Yellow corresponds to value 1 and blue corresponds to value 0.

As mentioned above, the procedure to convert a high genus surface to a genus zero surface is to first fill topological holes identified using medial axis, and then progressively increase the radius r of the α-shape till the genus becomes zero. The additional points that fill the holes identified by the medial axis are indexed 0, whereas the points on the original shape are indexed 1. A geometry image is created from the spherical parametrization by first projecting onto an octahedron, and then, the octahedron is cut to obtain a square grid. A geometry image is created by uniformly sampling the square domain and then interpolating the nearby feature values. This is explained in the code at the end of the supplementary material.

A geometry image, C_{idx} created from the points indexed 0 or 1 serves as a topological mask. However, the holes filled by increasing the radius r of the α-shape does not introduce additional points onto the shape, but instead introduces new faces. For such shapes, we calculate the point-wise Hausdorff distance between points in the geometry image (X,Y and Z coordinates) corresponding to the genus-zero shape and points on the original α-shape with r equal to the grid resolution. It is expected that the Hausdorff distance be small (or equal to 0) for a point in the geometry image which is contained in the original shape and be large for points lying on new faces in the transformed α-shape. Hence, we encode the Hausdorff distance, D, as a similarity value in the range [0, 1] such that values close to 0 indicate points on newly introduced faces and values close to 1 indicate points in the original shape. Mathematically:

$$C_{dist} = exp(-max((D-r),0)/\sigma) \tag{2}$$

The value of σ is set to r (r = 0.866 in all our experiments on the ModelNet benchmark as 0.866 corresponds to the radius of a sphere circumscribing a $1 \times 1 \times 1$ cube, i.e., $\sqrt{3}/2$). The topological mask is then calculated as:

$$C_{top} = C_{dist} \odot C_{idx} \tag{3}$$

Here C_{idx} corresponds to the geometry image of points on the shape indexed with values 0 or 1, and $\odot$ is the element wise Hadamard product. C_{top} then serves as a topological mask for the shape. The overall procedure to derive C_{top} for higher genus surfaces is to first calculate the medial axis and then calculate C_{idx} corresponding to the 0; 1 points (see Figure 1). If the genus of the resulting shape is 0 then C_{idx} directly serves as C_{top} for subsequent shape analysis. Else if the genus of the resulting shape is higher than 0, then we progressively increase r of the α-shape till the genus becomes zero and calculate C_{dist} (see Figure 2). C_{top} is then given by equation 3.

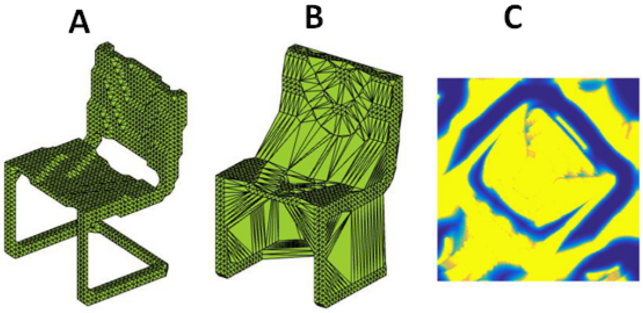

Fig. 2. A: Original α-shape corresponding to a chair with genus 2. Filling topological holes using the medial axis approach does not fill all topological holes of this shape. B:The resulting α-shape by increasing r from 0.866 to 8.14 which fills all topological holes. Observe no new points are added, however, new faces are introduced. C: The geometry image, C_{dist} created from distance of points, D on the geometry image to points on the original α-shape. Yellow corresponds to value 1 and blue corresponds to value 0.

2 Authalic parametrization and Laplacian

Our authalic parametrization is inspired by the technique of [6], although, we follow a different displacement strategy. We point the readers to Section 3 of [6] for a rigorous mathematical derivation of equations 2 to 6 in the main paper using Lie derivatives and Cartan's formula. However, our displacement strategy operates on the shape rather than the parametrization domain as done in [6]. This displacement strategy avoids (1) expensive re-triangulation, (2) re-calculating a Laplacian, (3) decomposing a vector field into components on a sphere, for each iteration. As a result the computation is a lot faster compared to [6] while being more accurate. Note that the elements of the discretized version of the Laplacian operator with cotangent weights are calculated as:

$$L(u,v) = \begin{cases} \sum_v \frac{cot a_{u,v} + cot b_{u,v}}{2} & \text{if}, v = u \\ -\frac{cot a_{u,v} + cot b_{u,v}}{2} & \text{if}, v \subseteq N_1(u) \\ 0 & \text{otherwise} \end{cases}$$

where $\mathcal{N}_1(u)$ is the set of 1-ring neighbours of vertex u, $a_{u,v}, b_{u,v}$ are the two angles supporting the edge connecting vertices u and v and E_B indicates a boundary edge. We assume that the triangulation is regular, i.e., $a_{u,v} + b_{u,v} \leq \pi$ so that all weights are positive.

3 CNN architecture

For the ModelNet benchmark, all shapes were voxelized in a grid with resolution $40 \times 40 \times 40$. This resolution was chosen because the number of points on surface of the cube is then less than 10000. Usually the surface of shapes in the ModelNet benchmark consisted of 2000-5000 vertices, a sufficient resolution to capture the key features in an authalic parametrization setting. The largest connected shape component was chosen from this grid, followed by the topological hole filling procedure described above. All geometry images used for evaluation are of size 64×64, unless mentioned otherwise. We list the structure of the CNN used for rigid and non-rigid shape analysis in Table 1. The training for rigid shape benchmarks converged after 20 epochs and for non-rigid shape benchmarks converged after 30 epochs. The learning rate was set at 0.001 for all our experiments. The 48-dimensional activation feature output from the penultimate layer was used for non-rigid shape retrieval and 96-dimensional activation feature output from the penultimate layer was used for rigid shape retrieval. We now state the timings for creating a geometry image for a mesh model with 2500 vertices. The authalic parametrization of the shape takes 8 seconds using Matlab. A geometry image tensor of size $64\times 64 \times 3 \times 12$ with three features: two principal curvatures and a topological mask and the 4^{th} dimension corresponding to the 12 views takes $\leq 0.5sec$. A geometry image tensor of size $64\times 64 \times 5 \times 12$ with five HKS features at logarithmically spaced times and the 4^{th} dimension corresponding to the 12 views takes $\leq 3sec$. We used MatConvNet (http://www.vlfeat.org/matconvnet/) for training the convolutional neural networks. The training times for ModelNet10 is 1 hour and memory requirement is 1.9 GB for storing the $64\times 64 \times 4 \times 58788$ tensor of geometry images at single precision. The training times for ModelNet40 is 3 hours and memory requirement is 4.1 GB for storing the $64\times 64 \times 4 \times 147732$ tensor of geometry images at single precision.

4 Parameter choice

There are three parameters for creating geometry images used in the convolutional neural nets: (1) Size of geometry image (2) Number of views or cuts by rotating the spherical parametrization about polar axis (3) Padding the geometry images. We discuss the effect of these parameters on classification accuracy for the ModelNet10 benchmark. Table 2 shows the classification accuracy of our approach using different sizes of geometry image. We see that although the classification accuracy slightly increases for larger sizes of geometry images, the training time for large geometry images such as that of 128×128 geometry image is significantly higher. The small increase in accuracy is natural as 3D objects are generally sparse in features compared to images. Hence, a geometry image of resolution 64×64 suffices to capture key attributes of a 3D shape. Table 3 shows the variation in classification accuracy for different number of views or cuts by rotating the spherical parametrization around a polar axis. The top row shows the accuracy by pooling predictions from the softmax layer over the number of views and then selecting the one with the highest overall score, whereas the bottom row shows the classification accuracy by only considering a single geometry image corresponding to the original spherical parametrization, i.e., without rotating around the polar axis,

	Layers	# Kernels	Filter size	Stride	Pad
1	Conv	16	5×5×4	1	0
2	Pmax			2	0
3	ReLU				
4	Conv	32	5×5×16	1	0
5	ReLU				
6	Pmax			2	0
7	Conv	48	4×4×32	1	0
8	ReLU				
9	Pmax			2	0
10	Conv	64	4×4×48	1	2
11	ReLU				
12	Pmax			2	0
13	Conv	96	1×1×64	1	0
14	ReLU				
15	Conv	10/40	1×1×96	1	0
16	Smax				

	Layers	# Kernels	Filter size	Stride	Pad
1	Conv	8	5×5×5	1	0
2	Pmax			4	0
3	ReLU				
4	Conv	12	4×4×8	1	0
5	Pmax			4	0
6	ReLU				
7	Conv	16	3×3×12	1	0
8	ReLU				
9	Conv	48	1×1×16	1	0
10	ReLU				
11	Conv	10/30	1×1×48	1	0
12	Smax				

Table 1. (Conv:convolutional layer, Pmax: max pooling layer, ReLU: rectified linear units layer, Smax: softmax layer) **Left** Architecture of CNN for rigid shape analysis, i.e., the ModelNet10 and ModelNet40 benchmarks. The input is a $64 \times 64 \times 4$ image corresponding to the two principal curvatures, the topological map and the height field. **Right** Architecture of CNN for non-rigid shape analysis, i.e., the McGill and SHREC benchmarks. The input is a $64 \times 64 \times 5$ image corresponding to HKS feature maps at 5 time stamps.

and tests the learning capacity of the CNN using data augmentation in training set and a single view in the testing set. We see that the classification accuracies naturally decrease by decreasing the number of cuts, however the accuracies differ by a small value as the number of cuts range from 6 to 12. This indicates that a moderate number of views suffices for the CNN to learn the rotational variance of the geometry images due to the cuts. This is further substantiated by the small variation in classification accuracy values for a single view geometry image in the testing phase. Finally table 4 shows the classification accuracy for different padding sizes. The size of the geometry image was kept constant, the padding are of different sizes as mentioned in the table, and the other values in the geometry image were imputed were zeros to keep the size constant at 64×64. The classification accuracies validate that padding increases accuracy of classification. Another point to note that is that no padding or full padding is better than partial padding. We also report the result for two other cases: (1) The classification accuracy by doing a multi-view CNN approach [4] for 6 views results in classification accuracy of 87.1 %, comparable to our classification accuracy of 87.6 for 6 views. We observed that the convergence of this approach using the implementation provided by [4] relied on heuristically tuning the learning rate and other parameters, and often failed to converge. This is possibly because [4] fine-tuned the ImageNet CNN, whereas we start from random weights. However, the benefit of the multi-view approach is that it results in a single shape descriptor covering all viewpoints. (2) The classification accuracy by considering three feature maps, i.e., considering the principal curvatures and the topological map without the height feature map is 88.3% indicating that our heuristic technique to align the polar axis works well in practice, with the height feature map providing marginal improvement in accuracy.

Value/Size	64×64	80×80	96×96	128×128
Accuracy	88.4	88.5	88.9	89.2
Training Time	52	60	110	360

Table 2. Classification accuracy and training time (in minutes) for different sizes of geometry image on the ModelNet10 dataset

Views	1	2	3	4	6	9	12
Sum pooling	74.1	80.2	84.4	86.8	87.6	87.9	88.4
Single view	74.1	80.8	83.7	84.9	86.1	87.5	88.0

Table 3. Classification accuracy for different number of views or number of cuts around polar axis for a spherical parametrization on the ModelNet10 dataset. The top row shows the accuracy for sum pooling over the number of views, and the bottom row shows the classification accuracy for a single view geometry image.

Padding	0×0	1×1	2×2	3×3	4×4
Accuracy	88.1	87.9	87.7	87.7	88.4

Table 4. Classification accuracy for different padding on a 56×56 geometry image to create a 64×64 geometry image on the ModelNet10 dataset

5 Parameter values for other methods

We use linear SVMs for classification task on non-rigid datasets. We use the same set of parameters in the openly available implementations for all methods, i.e., ShapeGoogle, Zerkine moments, Light Field Descriptor and 3DShapeNets. We briefly discuss the parameter values for these methods:

1. ShapeGoogle[1]: The HKS signatures used in ShapeGoogle was approximated using 100 eigenvectors and eigenvalues and computed at 5 time scales. ANN approach was used to perform clustering in the descriptor space and construct the vocabularies. The soft quantization parameter was set to twice the median size of the clusters in the geometric vocabulary [1].

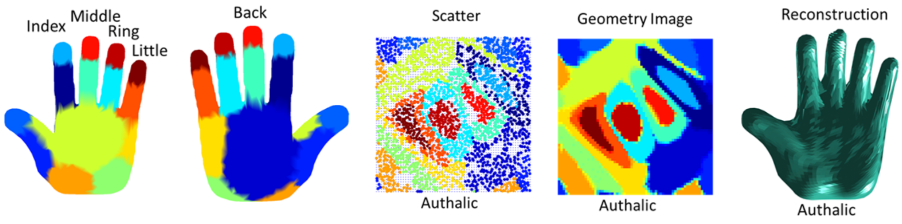

Fig. 3. Authalic spherical parameterization. Left to Right: The first two plots show that 2500 vertices of the hand mesh are color coded. These points are mapped onto a square domain using authalic spherical parametrization in the center plot. The 64× 64 geometry image is created by uniformly sampling the square domain (blue dots in the center plot), and then interpolating the nearby feature values. The authalic geometry image captures all tip features as shown in the second to last plot. This is validated by reconstructing shape from geometry images encoding X, Y, Z coordinates in the final plot.

2. Zerkine moments [3]: We used binary voxelization in a 64^3 grid and applied a kernel width of 2 and maximal moment order 20 to construct the Zerkine moments. The resulting Zerkine moment descriptor for a shape is a vector of 122 dimensions.
3. Light Field Descriptor [2]: We used the executable provided by the authors at `http://3d.csie.ntu.edu.tw/~dynamic/3DRetrieval/` for our experiments with light field descriptors. The parameters are the same as in the original paper.
4. 3DShapeNets [5]: We used the source code provided by the authors for our experiments with 3DShapeNets. The network architecture and parameters are the same as in the original paper.

6 Code for Geometry Image

Algorithm 1 is the pseudo-code for creating an authalic spherical parametrization of a mesh model as discussed in the main manuscript. We also provide a MATLAB implementation for creating a geometry image, given the spherical parametrization and feature map to be encoded. The process is visually described in Figure 3.

```
function gim = geometry_image(signal_original,pos_sphere, n)
% generate a geometry image from a spherical parameterization
% 'signal_original' is feature to be re-sampled on a regular grid
% 'pos_sphere' are spherical location for points
% 'n' is dimension of geometry image
% Adapted from wavelet toolbox of Gabriel Peyre

if nargin<3; n = 64; end
% compute sampling location on the image respecting spherical areas
posw = psps(pos_sphere);
% perform 4-fold symmetry
sym = { [0,1], [0,-1], [1, 0], [-1, 0] };
```

Algorithm 1 Authalic Spherical Parametrization

```
Input: $\mathcal{M} = (V, F, E)$, original mesh model, $n_{max}$, maximum number of iterations, $\epsilon$, threshold
Output: $V_s$, set of spherical vertex coordinates on spherical domain $S$
 1: Initialize $V_s$, using any spherical parametrization
 2: Compute $\hat{L} \leftarrow pinv(L)$, $L$ is cotangent Laplacian
 3: while $\epsilon < max_u(\delta h_u) \& iter < n_{max}$ do
 4:     Compute $\delta h_u \leftarrow \frac{A_u^s}{A_u} - 1$, the areal distortion
 5:     Compute $g \leftarrow \hat{L}\delta h_u$, the scalar field
 6:     Compute $\nabla g$, the gradient field at each face
 7:     Compute $\nabla g_u$, the gradient field at each vertex
 8:     Compute $v \leftarrow v + \rho\nabla g_v$, displacement on $M$
 9:     Perform $v_s \leftarrow bary(v)$, Barycentric mapping from $M$ to $S$
10:     Perform $v_s \leftarrow norm(v_s)$, normalize spherical coordinates
11: end while
12: return $(V_s)$ on $S$
```

```matlab
for i=1:length(sym)
    c = sym{i};
    if c(1)==0
        I = find(posw(2,:)*sign(c(2))>=0);
    else
        I = find(posw(1,:)*sign(c(1))>=0);
    end
    posi = posw(:,I); posi = perform_symmetry(posi,c);
    posw = [posw, posi];
    signal_original = [signal_original, signal_original(:,I)];
end
% cropping for speed
m = 1.5;
I = find( posw(1,:)<m & posw(1,:)>-m & posw(2,:)<m & posw(2,:)>-m);
posw = posw(:,I); signal_original = signal_original(:,I);
% remove doublons
[~,I] = unique(posw(1,:)+pi*posw(2,:)); posw = posw(:,I);
signal_original = signal_original(:,I);
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
% sampling location in [-1,1]
x = -1:2/(n-1):1; [Y,X] = meshgrid(x,x);
% interpolate location from triangles vertices to grid locations
gim = zeros( n, n, size(signal_original,1) );
for i=1:size(signal_original, 1)
    Fscatter = scatteredInterpolant(posw(1,:)',posw(2,:)',...
    signal_original(i,:)','natural');
    gim(:,:,i)=Fscatter(X,Y);
end
```

```
function y = perform_symmetry(x,c) % y = 2*c - x
y(1,:) = 2*c(1) - x(1,:); y(2,:) = 2*c(2) - x(2,:);

function posw = psps(pos_sphere)
% The sampling is first projected onto an octahedron and then...
% unfolded on a square.
% All 3-tuple of {-1,1}
a = [-1 1]; [X,Y,Z] = meshgrid(a,a,a);
anchor_2d = {}; anchor_3d = {};
for i=1:8
    x = X(i); y = Y(i); z = Z(i);
    a2d = []; a3d = [];
    if z>0
        a2d = [a2d, [0;0]];
    else
        a2d = [a2d, [x;y]];
    end
    a2d = [a2d, [x;0]]; a2d = [a2d, [0;y]];
    anchor_2d{i} = a2d;
    a3d = [a3d, [0;0;z]]; a3d =[a3d, [x;0;0]]; a3d =[a3d, [0;y;0]];
    anchor_3d{i} = a3d;
end
pos = pos_sphere; n = size(pos, 2); posw = zeros(2,n);
for s = 1:8
    x = X(s); y = Y(s); z = Z(s);
    anc2d = anchor_2d{s}; anc3d = anchor_3d{s};
    I = find( signe(pos(1,:))==x & signe(pos(2,:))==y ...
        & signe(pos(3,:))==z );
    posI = pos(:,I); nI = length(I);
    % find the area of the 3 small triangles
    p1 = repmat(anc3d(:,1), 1, nI);
    p2 = repmat(anc3d(:,2), 1, nI);
    p3 = repmat(anc3d(:,3), 1, nI);
    a1 = compute_spherical_area( posI, p2, p3 );
    a2 = compute_spherical_area( posI, p1, p3 );
    a3 = compute_spherical_area( posI, p1, p2 );
    % barycentric coordinates
    a = a1+a2+a3;
    % aa = compute_spherical_area( p1, p2, p3 );
    a1 = a1./a; a2 = a2./a; a3 = a3./a;
    posw(:,I) = anc2d(:,1)*a1 + anc2d(:,2)*a2 +anc2d(:,3)*a3;
end

function y = signe(x)
y = double(x>=0)*2-1;

function A = compute_spherical_area( p1, p2, p3 )
% length of the sides of the triangles :
% cos(a)=p1*p2
a = acos(dotp(p2,p3)); b = acos(dotp(p1,p3)); c = acos(dotp(p1,p2));
```

```
s = (a+b+c)/2;
% use L'Huilier's Theorem
E = tan(s/2).*tan((s-a)/2).*tan((s-b)/2).*tan((s-c)/2);
A = 4*atan(sqrt(E)); A = real(A);

function d = dotp(x,y)
d = x(1,:).*y(1,:) + x(2,:).*y(2,:) + x(3,:).*y(3,:);
```

References

1. A. M. Bronstein, M. M. Bronstein, L. J. Guibas, and M. Ovsjanikov. Shape google: Geometric words and expressions for invariant shape retrieval. *ACM Transactions on Graphics (TOG)*, 30(1):1, 2011. 6
2. D.-Y. Chen, X.-P. Tian, Y.-T. Shen, and M. Ouhyoung. On visual similarity based 3d model retrieval. *Computer Graphics Forum*, 22(3):223–232, 2003. 7
3. M. Novotni and R. Klein. Shape retrieval using 3d zernike descriptors. *Computer Aided Design*, 36:1047–1062, 2004. 7
4. H. Su, S. Maji, E. Kalogerakis, and E. G. Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proc. ICCV*, 2015. 5
5. Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao. 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1912–1920, 2015. 7
6. G. Zou, J. Hu, X. Gu, and J. Hua. Authalic parameterization of general surfaces using lie advection. *Visualization and Computer Graphics, IEEE Transactions on*, 17(12):2005–2014, 2011. 3