Neural Mesh Flow: 3D Manifold Mesh Generation via Diffeomorphic Flows

Kunal Gupta  Manmohan Chandraker  
University of California, San Diego  
{k5gupta, mkchandraker}@eng.ucsd.edu

Abstract

Meshes are important representations of physical 3D entities in the virtual world. Applications like rendering, simulations and 3D printing require meshes to be manifold so that they can interact with the world like the real objects they represent. Prior methods generate meshes with great geometric accuracy but poor manifoldness. In this work we propose Neural Mesh Flow (NMF) to generate two-manifold meshes for genus-0 shapes. Specifically, NMF is a shape auto-encoder consisting of several Neural Ordinary Differential Equation (NODE) blocks that learn accurate mesh geometry by progressively deforming a spherical mesh. Training NMF is simpler compared to state-of-the-art methods since it does not require any explicit mesh-based regularization. Our experiments demonstrate that NMF facilitates several applications such as single-view mesh reconstruction, global shape parameterization, texture mapping, shape deformation and correspondence. Importantly, we demonstrate that manifold meshes generated using NMF are better-suited for physically-based rendering and simulation. Code and data will be released\(^1\).

1 Introduction

Polygon meshes allow an efficient virtual representation of 3D objects, enabling applications in graphics rendering, simulations, modeling and manufacturing. Consequently, mesh generation or reconstruction from images or point sets has received significant recent attention. While prior approaches have primarily focused on obtaining geometrically accurate reconstructions, we posit that physically-based applications require meshes to also satisfy manifold properties. Intuitively, a mesh is manifold if it can be physically realized, for example, by 3D printing. Typically, reconstructed meshes are post-processed with humans in the loop for manifoldness, in order to enable ray tracing, slicing or Boolean operations. In contrast, we propose a novel deep network that directly generates manifold meshes (Fig. 5), alleviating the need for manual post-processing.

A manifold is a topological space that locally resembles Euclidean space in the neighbourhood of each point. A manifold mesh is a discretization of the manifold using a disjoint set of simple 2D polygons, such as triangles, which allows designing simulations, rendering and other manifold calculations. While a mesh data structure can simply be defined as a set \((V, E, F)\) of vertices \(V\) and corresponding edges \(E\) or face \(F\), not every mesh \((V, E, F)\) is manifold. Mathematically, we list various constraints on a singly connected mesh with the set \((V, E, F)\) that enables manifoldness\(^2\).

- Each edge \(e \in E\) is common to exactly 2 faces in \(F\) (Fig. 2a)
- Each vertex \(v \in V\) is shared by exactly one group of connected faces (Fig. 2b)
- Adjacent faces \(F_i, F_j\) have normals oriented in same direction (Fig. 2c)

\(^1\)https://kunalmgupta.github.io/projects/NeuralMeshflow.html
\(^2\)In the scope of this work, meshes do not exhibit defects like duplicate elements, isolated vertices, degenerate faces and inner surfaces that can also cause a mesh to be non-manifold.
In summary, we make the following contributions:

- We introduce NeuralMeshFlow, a novel approach for 3D shape generation that guarantees manifoldness of the generated mesh.
- Our method uses Neural ODEs for modeling the diffeomorphic flow, ensuring that the generated mesh is a manifold in the limit of infinitesimal small discretization.
- We propose novel architectural features such as an instance normalization layer that enables generating 3D shapes across multiple categories.
- Our approach ensures manifoldness of the generated mesh and guarantees it to be a manifold in the limit of infinitesimal small discretization.
- We show quantitative comparisons to prior works and highlight the importance of manifoldness in physically meaningful tasks such as rendering, simulation, and 3D printing.

Toy example: regularizer’s dilemma

Consider the task of deforming a template unit spherical mesh $S$ (Fig. 3a) into a target star mesh $T$ (Fig. 3b). We approximate the deformation with a multi-layer perceptron (MLP) $f_0$ with a hidden layer of 256 neurons with $\text{relu}$ and output layer with $\text{tanh}$ activation. We train $f_0$ by minimizing various losses over the points sampled from $S, T$. A conventional approach involves minimizing the Chamfer Distance $L_c$ between $S, T$, leading to accurate point predictions but several edge-intersections (Fig. 3c). By introducing edge length regularization $L_e$, we get fewer edge-intersections (Fig. 3d). But the solution is also geometrically sub-optimal. We can further reduce edge-intersections with Laplacian regularization (Fig. 3e), but this takes a bigger toll on geometric accuracy. Thus, attempting to reduce self-intersections by explicit regularization not only makes the optimization hard, but can also lead to predictions with lower geometric accuracy. In contrast, our proposed use of NODE (with dynamics $f_0$) is designed by construction [10, 9] to prevent self-intersections without explicit regularization (Fig. 3f).

In summary, we make the following contributions:
A novel approach to 3D mesh generation, Neural Mesh Flow (NMF), with a series of NODEs that learn to deform a template mesh (ellipsoid) into a target mesh with greater manifoldness.

Extensive comparisons to state-of-the-art mesh generation methods for physically based rendering and simulation (see supplementary video), highlighting the advantage of NMF’s manifoldness.

New metrics to evaluate manifoldness of 3D meshes and demonstration of applications to single-view reconstruction, 3D deformation, global parameterization and correspondence.

2 Related Work

Existing learning based mesh generation methods, while yielding impressive geometric accuracy, do not satisfy one or more manifoldness conditions (Fig. 5b). While indirect approaches [6, 7, 16–19] suffer from the non-manifoldness of the marching cube algorithm [20], direct methods [21, 4, 3, 5] are faced with the regularizer’s dilemma on the trade-off between geometric accuracy and higher manifoldness, illustrated in Fig. 3 and discussed in Sec. 1.

Indirect Mesh Prediction  Indirect approaches predict the 3D geometry as either a distribution of voxels [22–29], point clouds [7, 30] or an implicit function representing signed distance from the surface [17, 16, 19]. Both voxel and point set prediction methods struggle to generate high resolution outputs which later makes the iso-surface extraction tools ineffective or noisy [3]. Implicit methods feed a neural network with a latent code and a query point, encoding the spatial coordinates [17, 16, 19] or local features [31], to predict the TSDF value [17] or the binary occupancy of the point [16, 19]. However, these approaches are computationally expensive since in order to get a surface from the implicit function representation, several thousands of points must be sampled. Moreover, for shapes such as chairs that have thin structures, implicit methods often fail to produce a single connected component.

All the above methods depend on the marching cube algorithm [20] for iso-surface extraction. While marching cubes can be applied directly to voxel grids, point clouds first regress the iso-surface using surface normals. Implicit function representations must regress TSDF values per voxel and then perform extensive query to generate iso-surface based on a threshold \( \tau \). This is used to classify grid vertices \( v_i \in V \) as ‘inside’ (\( TSDF(v_i) \leq \tau \)) and ‘outside’ (\( TSDF(v_i) \geq \tau \)). For each voxel, based on the arrangement of its grid vertices, marching cubes [20, 32–34] follows a lookup-table to find a triangle arrangement. Since this rasterization of iso-surface is a purely local operation, it often leads to ambiguities [34, 32, 33], resulting in meshes being non-manifold.

Direct Mesh Prediction  A mesh based representation stores the surface information cheaply as list of vertices and faces that respectively define the geometric and topological information. Early methods of mesh generation relied on predicting the parameters of category based mesh models. While these methods output manifold meshes, they work only when parameterized manifold meshes are available for the object category. Recently, meshes have been successfully generated for a wide class of categories using topological priors [3, 4]. Deep networks are used to update the vertices of initial mesh to match that of the final mesh. AtlasNet [3] uses Chamfer distance applied on the vertices for training, while Pixel2Mesh [4] uses a coarse-to-fine deformation approach using vertex Chamfer loss. However, using a point set training scheme for meshes leads to severe topological issues and produced meshes are not manifold. Some recent works have proposed to use mesh regularizers like Laplacian [4, 5, 2, 35], edge lengths [2, 5] and normal consistency [2] to constrain the flexibility of vertex predictions, but they suffer from the regularizer’s dilemma discussed in Fig. 3, as better geometric accuracy comes at a cost of manifoldness.

In contrast to the above approaches, the proposed NMF achieves high resolution meshes with a high degree of manifoldness across a wide variety of shape categories. Similar to previous approaches [3–5], an initial ellipsoid is deformed by updating its vertices. However, instead of using explicit
We now introduce NeuralMeshFlow (Fig 4). The objective of conditional flow (NODE Block) therefore is to learn a mapping \( f \) that maps an initial condition \( x_0 \) to \( x_T \). The standard NODE formulation cannot be used directly for the task of 3D mesh generation since they lack any means to feed in shape embedding and are therefore restricted to learning a few shape. A naive way would be to concatenate features to point coordinates like is done with traditional MLPs \([4,5]\) but this destroys the shape regularizations properties due to several augmented dimensions\([10,9]\). Our key insight is that instead of a fixed NODE dynamics \( f_\theta \) we can use a family of dynamics \( f_{\theta|z} \) parameterized by \( z \) while still retaining the uniqueness property as long as \( z \) is held constant for the purpose of solving IVP with initial conditions \( \{x_0, x_T\} \).

The objective of conditional flow (NODE Block) therefore is to learn a mapping \( F_{\theta|z} \) (eq.1) given the shape embedding \( z \) and initial values \( \{(p^i_0, p^i)\) where \( M_1, M_0 \) are respectively the input and output point clouds.

\[
p^i_O = F_{\theta|z}(p^i_0, z) = p^i_0 + \int_0^T f_{\theta|z}(p^i, z) dt
\]  

**Instance Normalization.** Normalizing input and hidden features to zero mean and unit variance is important to reduce co-variate shift in deep networks \([39–44]\). While trying to deform a template mesh regularizers, NMF uses NODE blocks to learn the diffeomorphic flow to implicitly discourage self-intersections, maintain the topology and thereby achieve better manifoldness of generated shape. The method is end-to-end trainable without requiring any post-processing.

**3 Neural Mesh Flow**

We now introduce NeuralMeshFlow (Fig 4), which learns to auto-encode 3D shapes. NMF broadly consists of four components. First, the target shape \( M_T \) is encoded by uniformly sampling \( N \) points from its surface and feeding them to a PointNet\([36]\) encoder to get the global shape embedding \( z \) of size \( k \). Second, NODE blocks diffeomorphically flow the vertices of template sphere towards target shape conditioned on shape embedding \( z \). Third, the instance normalization layer performs non-uniform scaling of NODE output to ease cross-category training. Finally, refinement flows provide gradual improvement in quality. We start with a discussion of NODE and its regularizing property followed by details on each component.

**NODE Overview.** A NODE learns a transformation \( \phi_T : \mathcal{X} \to \mathcal{X} \) as solutions for initial value problem (IVP) of a parameterized ODE \( x_T = \phi_T(x_0) = x_0 + \int_0^T f_\theta(x_t) dt \). Here \( x_0, x_T \in \mathcal{X} \subset \mathbb{R}^n \) represent the input and output from the network respectively, while \( T \in \mathbb{R} \) is a hyper parameter which represents the duration of the flow from \( x_0 \) to \( x_T \). For a well behaved dynamics \( f_\theta \) (Lipschitz continuous) any two distinct trajectories of NODE can never intersect due to the existence and uniqueness of IVP solutions \([9,10]\). Moreover, NODE manifest the orientation preserving property of diffeomorphic flows\([12,11]\). These lead to strong implicit regularization against self-intersection and non-manifold faces. There are several other advantages to NODE compared to traditional MLPs such as improved robustness \([37]\), parameter efficiency \([1]\) and the ability to learn normalizing flows \([38,14,15]\) and homeomorphism \([10]\). We refer readers to \([9,10,13]\) for more details.

**Diffeomorphic Conditional Flow.** The standard NODE formulation cannot be used directly for the task of 3D mesh generation since they lack any means to feed in shape embedding and are therefore restricted to learning a few shape. A naive way would be to concatenate features to point coordinates like is done with traditional MLPs \([4,5]\) but this destroys the shape regularizations properties due to several augmented dimensions\([10,9]\). Our key insight is that instead of a fixed NODE dynamics \( f_\theta \) we can use a family of dynamics \( f_{\theta|z} \) parameterized by \( z \) while still retaining the uniqueness property as long as \( z \) is held constant for the purpose of solving IVP with initial conditions \( \{x_0, x_T\} \).
We compute chamfer distances

We therefore stack two NODE blocks in a sequence followed by an instance normalization layer

with poor geometric accuracy and manifoldness (Fig. 5a) without IN leads to self-intersections and non-manifold faces due to very complex dynamics being learnt. (e) A model trained without IN leads to self-intersections and non-manifold faces due to very complex dynamics being learnt. (e) A model with IN is smoother and regularized.

Figure 5: The impact of instance normalization (IN) and refinement flows in NMF. (a) Learning deformation of a template (black) to target shapes of different variances (red and green) require longer non-uniform NODE trajectories making learning difficult. (b) IN allows NODE to learn deformations to an arbitrary variance. This leads to simpler dynamics and can later be scaled back to correct shape variance. (d) A model trained with additional refinement. We report the geometric accuracy, manifoldness and inference time for different amounts of refinement in Table 5b. The reported quantities are averaged over the 11 Shapenet categories (this excludes watercraft and lamp where NMF struggles with thin structures). For details on per category ablation, please see the supplementary material. To summarize, the entire NMF pipeline can be seen as three successive diffeomorphic flows \{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2\} of the initial spherical mesh to gradually approach the final shape.

**Overall Architecture.** A single NODE block is often not sufficient to get desired quality of results. We therefore stack two NODE blocks in a sequence followed by an instance normalization layer and call the collection a deformation block. While a single deformation block is capable of achieving reasonable results (as shown by \mathcal{M}_{p0} in Fig. 4) we get further refinement in quality by having two additional deformation blocks. Notice how the \mathcal{M}_{p1} has a better geometric accuracy than \mathcal{M}_{p0} and \mathcal{M}_{p2} is sharper compared to \mathcal{M}_{p1} with additional refinement. We report the geometric accuracy, manifoldness and inference time for different amounts of refinement in Table 5b. The reported quantities are averaged over the 11 Shapenet categories (this excludes watercraft and lamp where NMF struggles with thin structures). For details on per category ablation, please see the supplementary material. To summarize, the entire NMF pipeline can be seen as three successive diffeomorphic flows \{\mathcal{F}_0, \mathcal{F}_1, \mathcal{F}_2\} of the initial spherical mesh to gradually approach the final shape.

**Loss Function.** In order to learn the parameters \( \Theta \) it is important to use a loss which meaningfully represents the difference between the predicted \( \mathcal{M}_p \) and the target \( \mathcal{M}_T \) meshes. To this end we use the bidirectional Chamfer Distance (eq. 2) on the points sampled differentiably\[^{45}\] from predicted \( \mathcal{M}_p \) and target \( \mathcal{M}_T \) meshes.

\[
L(\Theta) = \sum_{p \in \mathcal{M}_p} \min_{q \in \mathcal{M}_T} ||p-q||^2 + \sum_{q \in \mathcal{M}_T} \min_{p \in \mathcal{M}_p} ||p-q||^2
\]  

(2)

We compute chamfer distances \( L_{p1}, L_{p2} \) for meshes after deformation blocks \( \mathcal{F}_1, \mathcal{F}_2 \) of \( \mathcal{M}_p \). For meshes generated from \( \mathcal{F}_0 \) we found that computing chamfer distance \( L_0 \) on the vertices gave better results since it encourages predicted vertices to be more uniformly distributed (like points sampled from target mesh). We thus arrive at the overall loss function to train NMF.

\[
L = w_0 L_0 + w_1 L_{p1} + w_2 L_{p2}
\]

(3)

Here we take \( w_0 = 0.1, w_1 = 0.2, w_3 = 0.7 \) so as to enhance mesh prediction after each deformation block. The adjoint sensitivity \[^{46}\] method is employed to perform the reverse-mode differentiation through the ODE solver and therefore learn the network parameters \( \Theta \) using the standard gradient descent approaches.

**Dynamics Equation.** The Neural ODE \( \mathcal{F}_0 \) is built around the dynamics equation \( f_0 \) which is learned by a deep network. Given a point \( x \in \mathbb{R}^3 \) we first get 512 length point features by applying a linear layer. To condition the NODE on shape embedding, we extract a 512 length shape feature
from the shape embedding $z$ and multiply it element wise with the obtained point features to get the point-shape features. Thus, point-shape features contains both the point features as well as the global instance information. Lastly, we feed the point-shape features into two residual MLP blocks each of width 512 and subsequent MLP of width 512 which outputs the predicted point location $y \in \mathbb{R}^3$. Based on the findings of [47, 13] we make use of the tanh activation after adding the residual predictions at each step. This ensures maximum flexibility in the dynamics learned by the deep network. More details about the architecture can be found in supplementary material.

**Implementation Details** For auto-encoding, we uniformly sample $N = 2520$ from the target mesh and using PointNet[36] encoder, get a shape embedding $z$ of size $k = 1000$. During training, the Neural ODEs are solved with a tolerance of $1e^{-5}$ and interval of integration set to $t = 0.2$ for deforming an icosphere with 622 vertices. At test time, we use an icosphere of 2520 vertices and tolerance of $1e^{-5}$. We train NMF for 125 epochs using Adam [48] optimizer with a learning rate of $10^{-5}$ and a batch size of 250, on 5 NVIDIA 2080Ti GPUs for 2 days. For single view reconstruction, we train an image to point cloud predictor network with pretrained ResNet encoder of latent code size 1000 and a fully-connected decoder with size 1000,1000,3072 with relu non-linearities. The point predictor is trained for 125 epochs on the same split as NMF auto-encoder.

**4 Experiments**

In this section we show qualitative and quantitative results on the task of auto-encoding and single view reconstruction of 3D shapes with comparison against several state of the art baselines. In addition to these tasks, we also demonstrate several additional features and applications of our approach including latent space interpolation texture mapping, consistent correspondence and shape deformations in the supplementary material.

**Data** We evaluate our approach on the ShapeNet Core dataset [49], which consists of 3D models across 13 object categories. We use the training, validation and testing splits provided by [6] to be comparable to other baselines. We use rendered views from [6] and sample 3D points using [50].

**Evaluation criteria** We evaluate the predicted shape $M_p$ for geometric accuracy to the ground truth $M_T$ as well as for manifoldness. For geometric accuracy, we follow [2] and compute the bidirectional Chamfer distance according to (2) and normal consistency using (4) on 10000 points sampled from each mesh. Since Chamfer distance is sensitive to the size of meshes, we scale the meshes to lie within a unit radius sphere. Chamfer distances are report by multiplying with $10^3$. With $M_p$, $M_T$ the point sets sampled from $M_p$, $M_T$ and $\Lambda_{P,Q} = \{(p, argmin_q ||p - q||) : p \in P\}$, we define

$$L_n = |\tilde{M}_P|^{-1} \sum_{(p,q) \in \Lambda_{\tilde{M}_P, \tilde{M}_T}} |u_p \cdot u_q| + |\tilde{M}_T|^{-1} \sum_{(p,q) \in \Lambda_{\tilde{M}_T, \tilde{M}_P}} |u_q \cdot u_p| - 1 \quad (4)$$

We detect non-manifold vertices (Fig. 2(b)) and edges (Fig. 2(a)) using [51] and report the metrics ‘NM-vertices’, ‘NM-edges’ respectively as the ratio($\times 10^5$) of number of non-manifold vertices and edges to total number of vertices and edges in a mesh. To calculate non-manifold faces, we count number of times adjacent face normals have a negative inner product, then the metric ‘NM-Faces’ is reported as its ratio(%) to the number of edges in the mesh. To calculate the number of instances...
We report mean errors for shape generation from point clouds in Table 1 with 3 iterations of Laplacian smoothing. Further iterations of smoothing lead to loss of geometric accuracy which is similar in architecture to NMF but trained with a larger icosphere of 2520 vertices, leading to the best baseline proposed in AtlasNet. We qualitatively show the effects of non-manifoldness in almost 100 times that of MeshRCNN. We note that MeshRCNN suffers from the most number of non-manifold edges, the method has several non-manifold vertices and edges compared to deformation based methods due to the cubify step as part of the MeshRCNN pipeline which converts a voxel grid into a mesh, faces compared to the best baseline. NMF-M also gets the highest normal consistency performance. To this end, we create OccNet baselines OccNet-1, OccNet-2 and OccNet-3 with MISE upsampling of 1, 2 and 3 times respectively. For fair comparison to other baselines, we use OccNet’s refinement module to output its meshes with 5200 faces.

**Baselines** We compare with official implementations for Pixel2Mesh [4, 2], MeshRCNN [2] and AtlasNet [3]. We use pretrained models for all these baselines motioned in this paper since they share the same dataset split by [6]. We use the implementation of Pixel2Mesh provided by MeshRCNN, as it uses a deeper network that outperforms the original implementation. We also consider AtlasNet-O which is a baseline proposed in [3] that uses patches sampled from a spherical mesh, making it closer to our own choice of initial template mesh. We also create a baseline of our own called NMF-M, which is similar in architecture to NMF but trained with a larger icosphere of 2520 vertices, leading to slight differences in test time performance. To account for possible variation in manifoldness due to simple post processing techniques, we also report outputs of all mesh generation methods with 3 iterations of Laplacian smoothing. Further iterations of smoothing lead to loss of geometric accuracy without any substantial gain in manifoldness. We also compare with occupancy networks [16], a state-of-the-art indirect mesh generation method based on implicit surface representation, we compare with several variants of OccNet based on the resolution of Multi Iso-Surface Extraction algorithm [16]. To this end, we create OccNet baselines OccNet-1, OccNet-2 and OccNet-3 with MISE upsampling of 1, 2 and 3 times respectively. For fair comparison to other baselines, we use OccNet’s refinement module to output its meshes with 5200 faces.

**Auto-encoding 3D shapes** We now evaluate NMF’s ability to generate a shape given an input 3D point cloud and compare against AtlasNet and AtlasNet-O. We evaluate the geometric accuracy and manifoldness of generated meshes. Additionally, we show physically based renderings of generated meshes with dielectric and conductor materials to highlight artifacts due to non-manifoldness.

We report mean errors for shape generation from point clouds in Table 1, with per-category results in supplementary material. Notice that AtlasNet-O with smoothing has 20 times the self-intersection compared to NMF without any smoothing. With Laplacian smoothing, NMF becomes practically intersection free. With Laplacian smoothing, NMF becomes practically intersection free. NMF also outperforms the baselines with and without smoothing in terms of non-manifold faces. Since AtlasNet uses 25 mesh non-manifold open templates to construct the final mesh it yields a constant value of 7.40 for its non-manifold edges while AtlasNet-O and NMF have manifold-edges. All the three methods have manifold vertices. Finally, NMF generates meshes with a higher normal consistency, leading to more realistic results in simulations and physically-based rendering. Visualizations in Fig. 6 show severe self-intersections and flipped normals for AtlasNet baselines which are absent for NMF. This leads to NMF giving more realistic physically based rendering results. Note the reflection of red box and green ball through NMF mesh, which are either distorted or absent for AtlasNet. The blue ball’s reflection on conductor’s surface is closer to ground truth for NMF due to higher manifoldness. Please see supplementary for further visualizations.

**Single-view reconstruction** We evaluate NMF for single-view reconstruction and compare against state-of-the-art methods. Mean errors over ShapeNet categories are reported in Table 7, with per-category results in supplementary. We note significantly lower self-intersections for NMF compared to the best baseline even after smoothing. Our method again results in fewer than 50% non-manifold faces compared to the best baseline. NMF-M also gets the highest normal consistency performance. Due to the cubify step as part of the MeshRCNN pipeline which converts a voxel grid into a mesh, the method has several non-manifold vertices and edges compared to deformation based methods Pixel2Mesh, AtlasNet-O and NMF. AtlasNet suffers from the most number of non manifold edges, almost 100 times that of MeshRCNN. We note that MeshRCNN[2] better performance in Chamfer Distance come at a cost of other metrics. We qualitatively show the effects of non-manifoldness in Figure 7 and supplementary material. We observe that for dielectric material (second row), NMF is
Figure 7: Single View Reconstruction: We compare NMF to other mesh generating baselines for SVR. Top row shows mesh geometry along with self-intersections (red) and flipped normals (black). Physically based renders for dielectric and conductor material are shown in rows 2 and 3 respectively. Notice the reflection of checkerboard floor, occluded part of red box and balls are all visible through NMF render but not with other baselines. This is due to the presence of severe self-intersection and flipped normals. The reflection of blue ball on metallic table is more realistic for NMF than other methods.

Figure 8: Implicit Methods: OccNet fails to give meshes that are singly connected and MeshR-CNN has poor normal consistency along with severe self-intersections. Both OccNet and NMF has non-manifold edges (shown as zoomed out insets). NMF generates meshes that are visually appealing with higher manifoldness.

<table>
<thead>
<tr>
<th>Single View Recon.</th>
<th>Chamfer-L2 (#)</th>
<th>Normal Consistency (#)</th>
<th>NM-Vertices (#)</th>
<th>NM-Edges (#)</th>
<th>NM-Faces (#)</th>
<th>Self-Intersection (#)</th>
<th>Time (#)</th>
</tr>
</thead>
<tbody>
<tr>
<td>OccNet-1[8]</td>
<td>8.77</td>
<td>0.814</td>
<td>1.13</td>
<td>0.85</td>
<td>0.36</td>
<td>0.00</td>
<td>871</td>
</tr>
<tr>
<td>OccNet-2[8]</td>
<td>8.66</td>
<td>0.814</td>
<td>2.67</td>
<td>1.79</td>
<td>0.21</td>
<td>0.03</td>
<td>1637</td>
</tr>
<tr>
<td>OccNet-3[8]</td>
<td>8.33</td>
<td>0.814</td>
<td>2.79</td>
<td>1.90</td>
<td>0.15</td>
<td>0.09</td>
<td>6652</td>
</tr>
<tr>
<td>NMF</td>
<td>7.82</td>
<td>0.829</td>
<td>0.00</td>
<td>0.00</td>
<td>0.83</td>
<td>0.12</td>
<td>187</td>
</tr>
<tr>
<td>NMF w/ Laplace</td>
<td>8.64</td>
<td>0.837</td>
<td>0.00</td>
<td>0.00</td>
<td>0.45</td>
<td>0.00</td>
<td>292</td>
</tr>
</tbody>
</table>

Table 3: Comparison with Implicit Representation method

able to transmit background colors closest to the ground truth, whereas other baselines only reflect the white sky due to the presence of flipped normals.

Comparison with Implicit Representation method We evaluate NMF against state-of-the-art indirect mesh generation method OccNet for the task of single view reconstruction. Since Mean errors over ShapeNet categories are reported in Table 3 and qualitative results are shown in Fig 8. More details and physically based renderings can be found in the supplementary. We observe that NMF outperforms the best baseline OccNet-3 in terms of geometric accuracy. This is primarily because NMF predicts a singly connected mesh object as opposed to OccNet which leads to several disconnected meshes. Moreover, due to the limitations imposed by the marching cubes algorithm discussed in section 2, OccNet-1,2,3 have several non-manifold vertices and edges where as by construction, NMF doesn’t suffer from such limitation. An example of non-manifold edge is shown in figure 8. For sake of completeness, we also show the mesh generated by Mesh R-CNN that also suffers from non-manifold vertices and edges. NMF is also competitive with OccNet in terms of self-intersections since with Laplacian smoothing both methods practically become intersection free. While OccNet, outperforms NMF in terms of non-manifold faces, we argue that this comes at a cost of higher inference time. For reference, the fastest version of OccNet has comparable non-manifold faces and self-intersections but fares behind in terms of other metrics.

5 Conclusions

In this paper, we have considered the problem of generating manifold 3D meshes using point clouds or images as input. We define manifoldness properties that meshes must satisfy to be physically realizable and usable in practical applications such as rendering and simulations. We demonstrate that while prior works achieve high geometric accuracy, such manifoldness has previously not been sought or achieved. Our key insight is that manifoldness is conserved under a diffeomorphic flow
that deforms a template mesh to the target shape, which can be modeled by exploiting properties of neural ODEs. We design a novel architecture, termed Neural Mesh Flow, composed of deformation blocks with instance normalization and refinement flows, to achieve manifold meshes without any post-processing. Our results in the paper and supplementary material demonstrate the significant benefits of NMF for real-world applications.

**Broader Impact**

The broader positive impact of our work would be to inspire methods in computer graphics and associated industries such as gaming and animation, to generate meshes that require significantly less human intervention for rendering and simulation. The proposed NMF method addresses an important need that has not been adequately studied in a vast literature on 3D mesh generation. While NMF is a first step in addressing that need, it tends to produce meshes that are over-smooth (also reflected in other methods sometimes obtaining greater geometric accuracy), which might have potential negative impact in applications such as manufacturing. Our code, models and data will be publicly released to encourage further research in the community.

**Acknowledgement**

We would like to thank Krishna Murthy Jatavallabhula for valuable discussions. We would also like to thank Pengcheng Cao with UCSD CHEI for providing 3D printed models and Shreyam Natani for helping with Blender.

**References**


