Point2Mesh: A Self-Prior for Deformable Meshes Development Track

Byunghyun Kim
KAIST
rooknpown@kaist.ac.kr

Wonjung Park
KAIST
fabiola@kaist.ac.kr

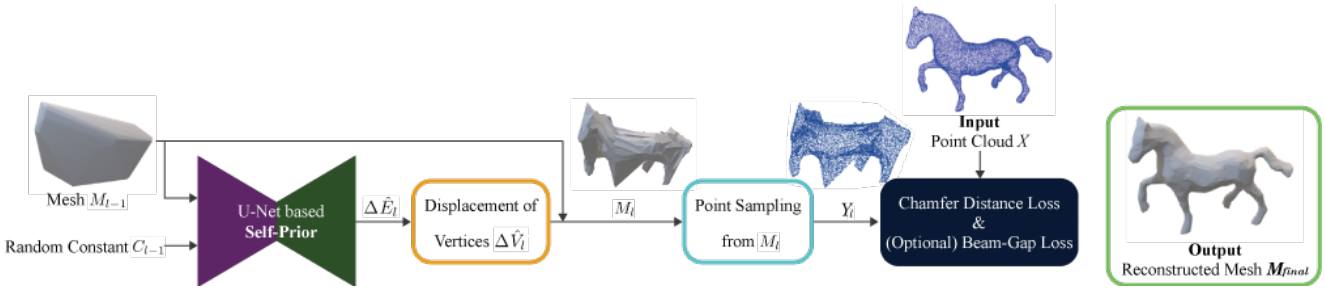

Figure 1. **Overview of Point2Mesh** Mesh M_{l-1} and fixed random noise vector C_{l-1} is fed into the U-Net [8] based Self-Prior network at lth iteration. Chamfer distance between input point cloud and point cloud from mesh M_l is computed as a loss function. The final output is generated by iterative mesh deformation. Horse point cloud is sampled from mesh data of COSEG [10]

Abstract

Point2Mesh [4] reconstructs triangular mesh from given point cloud with self-prior. Self-prior is a prior defined by its own point cloud which is trained inherently in the weights of neural networks. The initial mesh, convexly covering the given point cloud, is deformed by iteratively reconstructing the surface in more detail. Implementing self-prior network based on mesh network layers from MeshCNN [3] is the main task of this work. Bi-directional Chamfer loss and part of beam-gap loss are implemented for training. While most of the results match with the original paper qualitatively, reconstruction with similar number of faces show Poisson reconstruction outperforming Point2Mesh for some tasks quantitatively. Further ablation study on beam-gap loss is illustrated. Code is available at: https://github.com/rooknpown/Point2Mesh-CS492-A-.

1. Introduction

Surface reconstruction from point cloud to mesh is an important task since recent technologies on sensors such as LiDAR and light scanners obtain object surface information in the form of 3D point clouds while various applications in engineering, design, art require objects in form of mesh [1]. The original paper Point2Mesh [4] also focuses on this problem of reconstruction from point cloud to mesh, but with *self-prior*, where the neural network naturally learns the prior of the given single object.

The key challenge and the implementation focus is implementing self-prior network and two loss functions. A deep understanding on each network layers of MeshCNN [3] is required to build trainable self-prior in U-Net [8] structure. Second challenge is setting various hyperparameters for training including upsampling rate and maximum number of mesh faces. The author does not reveal the hyperparameters used for the results in the paper, so it is difficult to train optimally and achieve high accuracy.

While some input point cloud and initial mesh data are provided by the author, several data for the experiment were not obtained since the author did not reply to our contact. Instead, we acquire some mesh data from Thingi10k [11] and COSEG [10]. For training, it is important to keep the resolution of the data same as the baseline methods for fair comparison, but such details are not provided in the original paper. Therefore, we improve the the original work by reconstructing surface with similar number of faces compared to the baseline method, Poisson Reconstruction [6]. The F-score metric between input point cloud and points sampled from reconstructed mesh is used to evaluate the quality, as used in the original paper. While all reproducible experiments are tested, we also conduct additional ablation study not provided in the paper on beam-gap loss to show its effectiveness.

In summary, our achievements are:

- Implementing self-prior network, bi-directional Chamfer loss and part of beam-gap loss.
- Additional ablation study on beam-gap loss.
- Fair comparison with baseline method (Poisson reconstruction [6]) on a similar number of mesh faces.
- Processing data not provided by the author.
- Reproducing experiments provided in the paper.

2. Method

2.1. Problem setup

Given an input point cloud X, Point2Mesh reconstructs a triangular mesh. Starting with an initial watertight mesh M_0, it is iteratively deformed in order to match the input point cloud more closely. As illustrated in figure 1, at lth iteration, input mesh M_{l-1} is deformed to an output mesh M_l as the network predicts displacements of vertices $\Delta\hat{V}_l$ to move the mesh closer to the input point cloud. Chamfer distance between input point cloud and point cloud from mesh M_l is computed as a loss function.

2.2. Initial Mesh Approximation

Initial watertight mesh M_0 must be first obtained for deformation. The method for generating initial mesh depends on the topology of the object. For objects of genus-0 (topology with no hole), convex hull of the input point cloud is the initial mesh. For objects with genus greater than 0 alpha shape is first computed, then watertight manifold algorithm [5] with coarse octree resolution is applied to generate the initial mesh.

2.3. Self-Prior Network

Given the initial mesh M_0, the self-prior network should iteratively output displacements of vertices $\Delta\hat{V}_l$ from deformed mesh M_{l-1} at lth iteration. This is done by two steps. First, self-prior network takes two inputs mesh M_{l-1} and fixed random noise vector C_{l-1}, then outputs the displacements of each edges $\Delta\hat{E}_l$. Second, we calculate the displacements of each vertex $\Delta\hat{V}_l$. Since a vertex is connected to multiple edges, the displacement of the vertex is determined with the average of the estimated positions, calculated by the displacements of all connected edges.

The self-prior network is in U-Net [8] based structure consisting of mesh convolution layers and mesh pooling layers. The shared weights of CNN learns the local shape while the network is trained to form the global shape. Such behaviour meets with the self-similarity of objects in nature. It is also shown empirically that the convolution layers provides the self-prior inherently without any pre-defined priors.

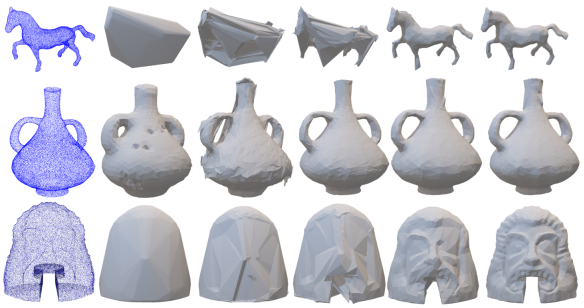

Figure 2. Horse and vase from COSEG [10] and spoonrider from Thingi10k [11] are applied to our framework. The point-cloud(first column) is sampled from the original mesh, and the initial mesh(second column) is obtained using alpha shape reconstruction with water tight manifold algorithm (for genus-2 vase) or convex hull (other genus-0 objects).

2.4. Loss Functions

2.4.1 Bi-directional Chamfer Distance

Bi-directional Chamfer distance is used as the main loss function to calculate the distance between the input point cloud X and the sampled points $\hat{Y}$ from the deformed mesh. Bi-directional Chamfer distance is defined as:

$$d(X,\hat{Y}) = \sum_{x\in X}\min_{\hat{y}\in\hat{Y}}\|x-\hat{y}\|_2 + \sum_{\hat{y}\in\hat{Y}}\min_{x\in X}\|x-\hat{y}\|_2.$$

2.4.2 Beam-Gap Loss

Using only Bi-directional Chamfer distance is not enough to obtain accurate mesh for objects with narrow or deep holes, and may result in local minimum. Therefore, additional beam-gap loss is utilized. Beam-gap is calculated by the beam rays from the mesh to the input point cloud. More precisely, from the point $\hat{y}$ sampled from the mesh surface, a beam is shot to the direction of the normal as cylinder with radius ϵ. Let the point in the input pointcloud that overlaps with the beam cylinder and is closest to $\hat{y}$ be $\mathcal{B}(\hat{y})$. Then, the beam-gap loss is defined as:

$$\mathcal{L}_b(X,\hat{Y}) = \sum_{\hat{y}\in\hat{Y}}\|\hat{y}-\mathcal{B}(\hat{y})\|^2.$$

If there is no collision to any points in the input pointcloud, $\mathcal{B}(\hat{y}) = \hat{y}$ so the loss is 0.

3. Implementation Details

3.1. Mesh and Mesh Network Layers

Mesh structure is required to operate on its edges, vertices, faces. In addition to the basic mesh class, submesh (PartMesh in the original paper) is implemented to

divide mesh into parts so that it fits in the GPU memory. MeshCNN [3] presents convolution and pooling layers for mesh structure. The internal calculations are too complex and does not have many ways to implement, so we decided to use them with some modifications.

3.2. Self-Prior Network and Loss Functions

We implemented U-Net-based [8] self-prior network harnessing mesh structure and mesh network layers provided from MeshCNN [3]. At the encoder side convolution and pooling decreases the dimension, and at the decoder side up-convolution and unpooling are operated. The convolution and up-convolution are implemented based on the MeshConv, and mesh pooling from MeshPool and MeshUnpool layers of MeshCNN [3].

The two loss functions, bi-directional Chamfer loss and part of beam-gap loss are implemented using knn functions from pytorch3d. Handling data structures and loss calculation are implemented by ourselves. However, we utilize original code for projection part to compute Beam-gap loss.

3.3. Data Processing and Training

Points are sampled using pytorch3d functions from meshes in Thingi10k [11] and COSEG [10] to get the input point cloud X. Initial mesh M_0 is obtained by convex hull function of Open3D library. For objects with genus greater than 0, alpha shape is created with Open3D function and watertight manifold is created with [5].

Necessary number of submeshes are created from the initial mesh depending on the number of faces to fit in GPU memory. Then, the self-prior network weights and required data structures are initialized. For each iteration l, the input mesh M_{l-1} and random noise vector C_{l-1} is fed into the self-prior network. For each submesh, the network calculates the updated vertices. The edges and faces are computed correspondingly. Points and normals are sampled from this deformed mesh, then loss is calculated with the input X and backpropagated. Bi-directional Chamfer loss is the default loss, and beam-gap loss is used alternately with bi-directional Chamfer loss for objects with cavities.

4. Experimental Results

4.1. Point clouds with non-ideal conditions

The power of self-prior CNN is to encode entire shape and inherently remove outliers regarding global features extracted by CNN. Thus, self-prior is known to be more robust to the point clouds with non-ideal conditions. Qualitatively, Point2Mesh shows powerful reconstruction on unoriented normals and real scan point clouds compared to Poisson reconstruction [6] as elucidated in the sections 4.1.2, 4.1.3. Also, Point2Mesh shows better reconstruction quality on point cloud with noise and missing regions than Poisson method [6]; however, Poisson outperforms ours and author's Point2Mesh in 4.1.1 quantitatively.

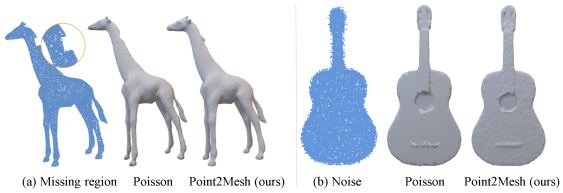

Figure 3. Effects of self-prior (a) Missing region is filled with Point2Mesh. (b) Point2Mesh has better denoising effect for noisy point cloud.

4.1.1 Point clouds with noise and empty regions

Four different point clouds of 25000 points from [4] are prepared. Small Gaussian noise is added to Tiki and Guitar to generate noisy point clouds, and portion of points are removed from Bull and Giraffe to test on point clouds with missing regions. To provide fair comparison, we first apply Poisson reconstruction [6] with depth 7 and count the number of faces. The counted number of faces is also set as the number of faces of Point2Mesh reconstruction. For the evaluation metric, the F-score [7] is measured to check the reconstruction quality as used in the original paper.

As elucidated in table 1, Poisson [6] has better F-score than Point2Mesh with same number of faces. This is because Point2Mesh has some margins between points and mesh when it is not fully converged, where F-score measures the distance between point cloud and mesh. However, as depicted in figure 3, since self-prior learns global features of the point clouds, Point2Mesh has better visual quality.

		Poisson	Ours	Author's
Noise	Tiki	94.7 ± 0.04	0.1 ± 0.1	1.1 ± 2.0
	Guitar	73.3 ± 0.6	4.7 ± 2.1	3.6 ± 0.8
Empty	Bull	65.4 ± 0.4	3.6 ± 1.8	4.9 ± 4.2
	Giraffe	93.8 ± 0.3	15.3 ± 4.5	15.5 ± 2.0

Table 1. F-score(%) between the input point cloud and reconstructed meshes by Poisson reconstruction method [6], our Point2Mesh and orginal author's Point2Mesh. The result is derived by 5 different noisy point clouds for Tiki and Guitar, and 5 different point clouds with missing regions for Bull and Giraffe. Larger F-score means that mesh is closer to the input point cloud.

4.1.2 Point clouds with unoriented normals

Poisson reconstruction [6] assumes oriented normals, thus cannot cope with point clouds with unoriented normals. On the other hand, as depicted in figure 4, Point2Mesh is more robust for artificially unoriented normals since Chamfer distance loss imposes relatively small penalty for normal misalignment than position.

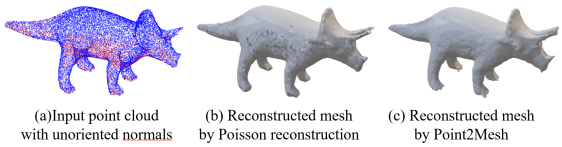

Figure 4. (a) Gaussian noise on normals of Red color points from is added. (b) Poisson [6] outputs uneven result while our Point2Mesh shows smooth mesh regardless of the noisy normals. The point cloud is from [4].

4.1.3 Real Scans

We compare the reconstruction of real scanned point clouds from [2] with Poisson reconstruction [6]. As depicted in 5, our Point2Mesh has better results than Poisson since real scans have noise and empty region of point clouds. Also, real scans does not guarantee correct orientation of normals. Therefore, Poisson method, assuming the oriented normals, has poor results. The point clouds are manually extracted from the real scan and has empty areas especially on the bottom and back parts.

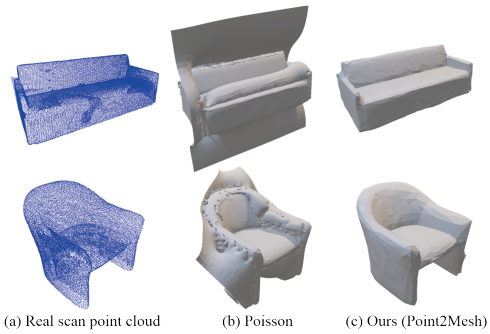

Figure 5. From real scan [2] sofa and chair point clouds (a), meshes are reconstructed by (b) Possion [6] and (c) Point2Mesh. Point2Mesh provides robust reconstruction especially for bottom and back parts where points are not scanned.

4.2. Smoothness-prior vs Self-prior

Smoothness prior is proposed by Sharf et al. [9] to avoid local minima when reconstructing a mesh. On the other hand, Point2Mesh suggests a self-prior which is automatically learned from a single input point cloud by a CNN instead of the explicit prior. Since the self-prior CNN inherently learns global features of the point cloud, it shows better reconstruction for point clouds with missing part by remaining global shape as depicted in figure 6. As figure 6 (a) shown, the original paper shows self-prior is powerful to remain bulging shape in empty region. Since the point clouds is not open source, a similar input is created as illustrated in 6 (b). The result still shows self-prior of Point2Mesh keep the sphere shape of the empty region.

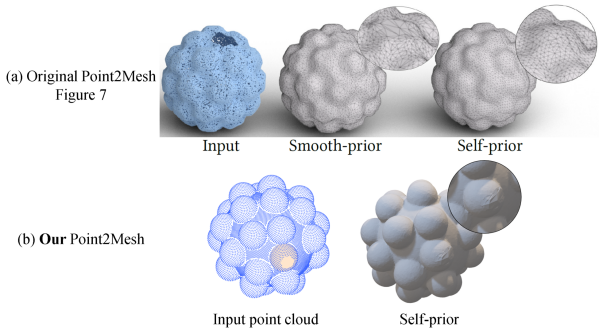

Figure 6. (a) is from the original paper point2mesh [4]. We create our similar point clouds in (b) with empty space since the point cloud is not offered. Still, our network remains the global features of point cloud for the empty region.

4.3. Ablation Study on Beam-Gap Loss

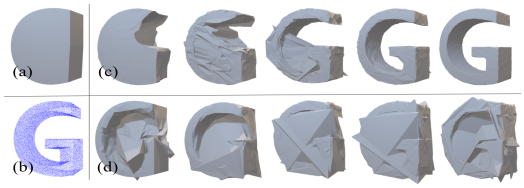

Figure 7. (a) is the initial mesh, and (b) is the input pointcloud from [4]. (c) with beam-gap loss successfully reconstruct the surface, whereas (d) without beam-gap loss fails to 'dig' the cavity.

There is no ablation study on beam-gap loss shown in the original paper, so it is conducted on G shape as shown in 7. The shape is reconstructed well with beam-gap loss by digging into the cavity, but it remains in local minimum with only bi-directional Chamfer loss.

5. Conclusion

Our implementation on self-prior network and loss functions were able to reproduce most of the results in the original paper. Qualitative results were similar to the original paper, while quantitative results of 4.1.1 showed worse performance than Poisson reconstruction on similar resolution. The ablation study on beam-gap loss showed its effectiveness on objects with deep cavities.

Acknowledgments

Byunghyun Kim: Implementing self-prior network and the two loss functions. Writing part of report.

Wonjung Park: Implementing data processing for initial mesh generation and ablation studies with Open3D library(open3d.org). Writing part of report.

Borrowed data from Thingi10k [11], COSEG [10], Point2Mesh [4], A Large Dataset of Object Scans [2]

Borrowed code for watertight manifold [5] mesh [4] and MeshCNN [3]. Check our github for specific links.

References

[1] Matthew Berger, Andrea Tagliasacchi, Lee M Seversky, Pierre Alliez, Gael Guennebaud, Joshua A Levine, Andrei Sharf, and Claudio T Silva. A survey of surface reconstruction from point clouds. In *Computer Graphics Forum*, volume 36, pages 301–329. Wiley Online Library, 2017. 1

[2] Sungjoon Choi, Qian-Yi Zhou, Stephen Miller, and Vladlen Koltun. A large dataset of object scans. *arXiv preprint arXiv:1602.02481*, 2016. 4

[3] 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. 1, 3, 4

[4] Rana Hanocka, Gal Metzer, Raja Giryes, and Daniel Cohen-Or. Point2mesh. *ACM Transactions on Graphics*, 39(4), aug 2020. 1, 3, 4

[5] Jingwei Huang, Hao Su, and Leonidas Guibas. Robust watertight manifold surface generation method for shapenet models, 2018. 2, 3, 4

[6] Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proceedings of the fourth Eurographics symposium on Geometry processing*, volume 7, 2006. 1, 2, 3, 4

[7] Arno Knapitsch, Jaesik Park, Qian-Yi Zhou, and Vladlen Koltun. Tanks and temples: Benchmarking large-scale scene reconstruction. *ACM Transactions on Graphics (ToG)*, 36(4):1–13, 2017. 3

[8] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer, 2015. 1, 2, 3

[9] Andrei Sharf, Thomas Lewiner, Ariel Shamir, Leif Kobbelt, and Daniel Cohen-Or. Competing fronts for coarse–to–fine surface reconstruction. In *Computer Graphics Forum*, volume 25, pages 389–398. Wiley Online Library, 2006. 4

[10] Yunhai Wang, Shmulik Asafi, Oliver Van Kaick, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. Active co-analysis of a set of shapes. *ACM Transactions on Graphics (TOG)*, 31(6):1–10, 2012. 1, 2, 3, 4

[11] Qingnan Zhou and Alec Jacobson. Thingi10k: A dataset of 10,000 3d-printing models, 2016. 1, 2, 3, 4