Supplement: SpaceMesh: A Continuous Representation for Learning Manifold Surface Meshes

In the supplement, we provide additional experimental details for mesh fitting (Section 1 and Section 2) and learning experiment (Section 3). We further provide methodology details for applying our method to mesh repair tasks in Section 4.

1 SINGLE MESH FITTING

To fit a single mesh using our SpaceMesh representation, we optimize the per-vertex embeddings $(x_i, y_i^{\text{root}}, y_i^{\text{prev}}, y_i^{\text{next}})$. The adjacency embeddings, x_i, are set to a dimension of 16, while each permutation embedding has a dimension of 6. The same dimensions for space and time coordinates are used across all experiments, with $k^s = k^t = k/2$. During training, the permutation matrix Φ is constructed using the ground truth adjacency matrix. The regularization parameter λ in Eq. 3 of the main paper is set to $\lambda = 4\frac{N_{\text{edges}}}{{N_{\text{vertices}}}^2}$, where N_{edges} and N_{vertices} represent the number of edges and vertices in each shape, respectively.

The Adam optimizer [Kingma and Ba 2014] is employed with a learning rate of 0.1. The overfitting process typically converges within 70 iterations. For comparison with DMesh [Son et al. 2024], we used the official released code[1]. For comparison with DSE [Rakotosaona et al. 2021], we used the official released code[2] and trained the networks to overfit a single mesh each time.

2 FITTING COLLECTIONS OF MESHES

In this experiment we follow a typical autodecoder setup, optimizing one 512-dimension latent code for each mesh in the dataset and employing a Transformer [Vaswani et al. 2017] to decode each latent code into the corresponding mesh. Specifically, for each mesh, the latent code is repeated N times (where N is the maximum number of vertices in the dataset), and a learnable positional embedding (shared across different meshes) is added to the repeated latent code. This resulting tensor is then passed to the Transformer to predict the vertex positions and per-vertex embeddings. To accommodate varying numbers of vertices across different meshes, all mesh vertices are padded with zeros up to $N = 2000$.

Additionally, a mask channel is appended to the vertex positions, with ground truth values of -1 or 1 indicating whether the vertex is padded or not, respectively. During training, the latent code for each shape, the positional embedding, and the Transformer weights are jointly optimized using a combination of the L2 loss on the predicted vertices and the losses described in Section 3. During inference, vertices with negative predictions in the mask channel are pruned.

Consistent with the overfitting experiment, the regularization parameter λ in Eq. 3 is set to $\lambda = 4\frac{N_{\text{edges}}}{{N_{\text{vertices}}}^2}$, and a learning rate of 0.001 is used.

[1]https://github.com/SonSang/dmesh
[2]https://github.com/mrakotosaon/dse-meshing

Author's address:

Dataset. We conduct our experiments on the Thingi10K [Zhou and Jacobson 2016] dataset, which has a diverse collection of real-world 3D printing models exhibiting a variety of shape complexities, topologies, and discretizations. From this dataset, we filter a subset consisting of manifold meshes with vertex counts ranging between 1000 and 2000, and randomly select 200 meshes. For each selected mesh, the vertices are sorted in z-y-x order, in accordance with the methodology of PolyGen [Nash et al. 2020].

3 LEARNING TO MESH THE SHAPE

3.1 Training Details

In this experiment, we use a dimension of 32 for adjacency embeddings and a dimension of 12 for each permutation embedding. The total number of channels for the output of the vertex connectivity prediction network is 68.

For the point cloud encoder, we use a PVCNN [Liu et al. 2019] with 4 PVConv layers, each with voxel resolutions of 32, 16, 8, and channels of 64, 128, and 256, respectively. The vertex position generation network follows the transformer-based diffusion model as in the original Point-E [Nichol et al. 2022], with 10 residual self-attention blocks of width equals 256. As mentioned in the main paper, at each denoising step, we concatenate the vertices' positions with features that are tri-linearly interpolated with the multi-resolution feature volumes from the encoder. The concatenated features are further passed through 4 PVConv layers with the same dimensions as the point cloud encoder. The training follows the standard diffusion model scheme with 1024 diffusion timesteps.

For the vertex connectivity prediction model, we also use PVConv to process the interpolated features. During training, our vertex generation model, which is based on a set transformer, does not hold correspondence between denoised vertices and ground truth vertices. Consequently, we use the ground truth vertices as input for training the connectivity prediction model, allowing supervision by the ground truth connectivity. We observe that the vertex generation model converges more quickly than the connectivity prediction network. Therefore, we train the connectivity prediction network for 10 steps for every single training step of the vertex generation model.

For the ABC dataset, since all meshes have the same number of vertices, we do not additionally predict a vertex mask. For the ShapeNet dataset, we predict 512 vertices, which is the maximum number of vertices in the filtered dataset, and an additional mask channel. We train our model using the Adam optimizer with a learning rate of 0.0001 for 800k iterations until convergence. To combine the loss for vertex prediction and connectivity prediction, we multiply the loss function from Equation. 3 by 200 and add it to the loss in Equation. 5 and the diffusion loss, both with a scale of 1.

Evaluation Metrics. We briefly describe the metrics used to evaluate the reconstruction quality. In all experiments, the longest dimension of all meshes is normalized to [-1, 1].

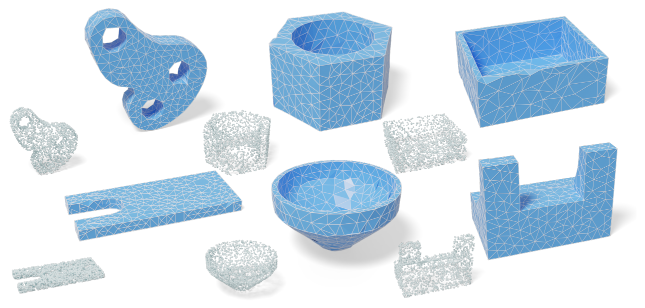

Fig. 1. Results on the ABC dataset generated by our learning to mesh model.

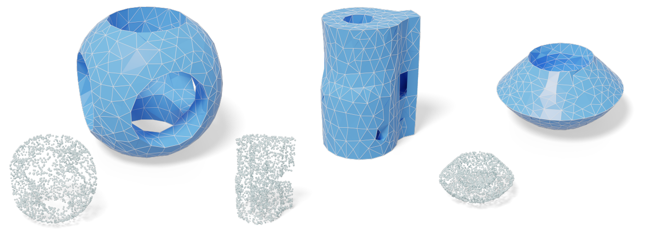

Fig. 2. Results on the Thingi10k dataset generated by our learning to mesh model trained on ABC dataset.

Chamfer Distance (CD) measures the distance between two point clouds using nearest neighbor search, sampling 10,000 points from the surface of each mesh. The F1-score is computed for the same point sets used for CD. Precision is determined by classifying points on the predicted mesh as true positives if their distance to the nearest ground truth (GT) point cloud is less than 0.02; otherwise, they are false positives. Recall uses the same 0.02 threshold. Edge Chamfer Distance (ECD) and Edge F-score (EF1) evaluate the reconstruction of sharp features, following prior works [Chen et al. 2022]. Each point in the sampled point cloud is checked by comparing the dot products between its normal and those of its neighbors; if the mean dot product is below 0.2, the point is classified as an edge point. ECD and EF1 then measure the Chamfer Distance and F1-score between these edge points. For the percentage of inaccurate normals, we use a 10-degree threshold. The normal of a point sampled from the reconstructed mesh is considered as inaccurate if the angles between its normals and that of the nearest ground truth point exceed 10 degrees.

3.2 Additional Results

Results on ABC Dataset. We present additional qualitative results produced by our learning-to-mesh model, trained on the ABC dataset, in Figure 1. To further evaluate our model, we perform stress tests by meshing novel shapes from the Thingi10k dataset [Zhou and Jacobson 2016], with qualitative results shown in Figure 2. The results demonstrate that our model accurately reconstructs 3D manifold meshes from input point clouds, showcasing its ability to generalize to unseen shapes during training.

4 DOWNSTREAM TASK: MESH REPAIR

We provide details on the algorithm to repair a partial mesh using our method. We first obtain the geometry context by sampling the point cloud on the whole surface and feeding it into the point cloud encoder. To generate vertex positions on the target region, while maintaining the original vertices on the untouched region, we redesign the vertex sampling process in the vertex diffusion model, following standard diffusion in-painting approaches.

Specifically, after removing the region to be repaired from the original mesh, we denote the partial mesh that we want to complete as $\mathcal{M}^{\text{known}} = (\mathcal{V}^{\text{known}}, \mathcal{E}^{\text{known}}, \mathcal{F}^{\text{known}})$. To begin the denoising process, we sample *all* vertices from a Gaussian distribution for time step T: $\mathcal{V}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. At each denoising step t, we first align $\mathcal{V}_t$ with the known vertices $\mathcal{V}^{\text{known}}$ to obtain a mask $\mathbf{m}$ which takes value 1 if the vertex is in $\mathcal{V}_t$ and 0 otherwise — see paragraph below. We feed $\mathcal{V}_t$ into our point cloud diffusion model: $\mathcal{V}_{t-1}^{\text{unknown}} \sim \mathcal{N}(\mu_\theta(\mathcal{V}_t, t), \Sigma_\theta(\mathcal{V}_t, t))$, where $\mu_\theta, \Sigma_\theta$ is the mean and variance prediction from our model, respectively. The denoised vertices will be $\mathcal{V}_{t-1} = \mathbf{m} \odot \mathcal{V}_{t-1}^{\text{known}} + (1 - \mathbf{m}) \odot \mathcal{V}_{t-1}^{\text{unknown}}$, where $\mathcal{V}_{t-1}^{\text{known}} \sim \mathcal{N}\left(\sqrt{\bar{\alpha}_t}\mathcal{V}^{\text{known}}, (1 - \bar{\alpha}_t)\,\mathbf{I}\right)$. After sufficiently many denoising steps we have the final vertices $\mathcal{V}_0$ that match the original mesh except in the region that has been repaired.

The new connectivity is predicted using our connectivity generation model. In this case, to preserve the existing edges and faces from the partial mesh, we only altered the connectivity between the predicted vertices and the boundary vertices from $\mathcal{V}^{\text{known}}$.

Aligning $\mathcal{V}^{known}$ and $\mathcal{V}_t$. Since it is challenging to align two point sets with different numbers of points for each set, we first append surface points sampled from the masked regions, denoted as p^{masked}, to the known vertices $\mathcal{V}^{\text{known}}$, such that $|\mathcal{V}_t| = |\mathcal{V}^{\text{known}}|$. We then solve the correspondence by first computing a cost matrix $\mathbf{C}$, where each item in the matrix $\mathbf{C}_{ij} = \|\mathcal{V}_{t,j} - \mathcal{V}_i^{\text{known}}\|_2^2$. To determine the one-to-one correspondence, we adopt the same strategy that we use for determining the local ordering (Section 3.3 in the main paper). In short, we apply Sinkhorn normalization [Sinkhorn 1964] to recover a doubly-stochastic matrix, $\hat{\mathbf{C}}$, representing a *softened* permutation matrix [Adams and Zemel 2011]. The correspondence can be recovered by computing the optimal unconstrained lowest-cost matching [Jonker and Volgenant 1988].

REFERENCES

Ryan Prescott Adams and Richard S Zemel. 2011. Ranking via sinkhorn propagation. *arXiv preprint arXiv:1106.1925* (2011).

Zhiqin Chen, Andrea Tagliasacchi, Thomas Funkhouser, and Hao Zhang. 2022. Neural Dual Contouring. *ACM Transactions on Graphics (Special Issue of SIGGRAPH)* 41, 4 (2022).

Roy Jonker and Ton Volgenant. 1988. A shortest augmenting path algorithm for dense and sparse linear assignment problems. In *DGOR/NSOR: Papers of the 16th Annual Meeting of DGOR in Cooperation with NSOR/Vorträge der 16. Jahrestagung der DGOR zusammen mit der NSOR.* Springer, 622–622.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. 2019. Point-voxel cnn for efficient 3d deep learning. *Advances in neural information processing systems* 32 (2019).

Charlie Nash, Yaroslav Ganin, SM Ali Eslami, and Peter Battaglia. 2020. Polygen: An autoregressive generative model of 3d meshes. In *International conference on machine learning*. PMLR, 7220–7229.

Alex Nichol, Heewoo Jun, Prafulla Dhariwal, Pamela Mishkin, and Mark Chen. 2022. Point-e: A system for generating 3d point clouds from complex prompts. *arXiv preprint arXiv:2212.08751* (2022).

Marie-Julie Rakotosaona, Paul Guerrero, Noam Aigerman, Niloy J Mitra, and Maks Ovsjanikov. 2021. Learning delaunay surface elements for mesh reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.* 22–31.

Richard Sinkhorn. 1964. A relationship between arbitrary positive matrices and doubly stochastic matrices. *The annals of mathematical statistics* 35, 2 (1964), 876–879.

Sanghyun Son, Matheus Gadelha, Yang Zhou, Zexiang Xu, Ming C Lin, and Yi Zhou. 2024. DMesh: A Differentiable Representation for General Meshes. *arXiv preprint arXiv:2404.13445* (2024).

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

Qingnan Zhou and Alec Jacobson. 2016. Thingi10k: A dataset of 10,000 3d-printing models. *arXiv preprint arXiv:1605.04797* (2016).