Point2Skeleton: Learning Skeletal Representations from Point Clouds Supplementary Material

1. Network Architecture

Fig. 1 shows the detailed network architecture of our method. We use PointNet++ [7] as the encoder of the input point cloud, which is composed of 4 set abstraction (SA) levels. For each SA, we show the radius of the ball query, the number of local patches and the feature dimensions of the MLPs. We adopt a density adaptive strategy, i.e., multi-scale grouping (MSG), to combine the features from two different scales in each layer. The shared MLPs are for processing the contextual features encoded by the PointNet++ to predict the final combinational weights, where each layer is followed by a batch normalization and a ReLU non-linearity. We also use a dropout layer with a rate of 0.2 during training.

The contextual features of the input points are linearly combined using the predicted convex combination weights which serve as the input surface point features corresponding to a skeletal point. The combined contextual features are concatenated with the information of skeletal spheres (center coordinates and radii) to serve as the node features that are input to the GAE for link prediction. Each GCN layer is

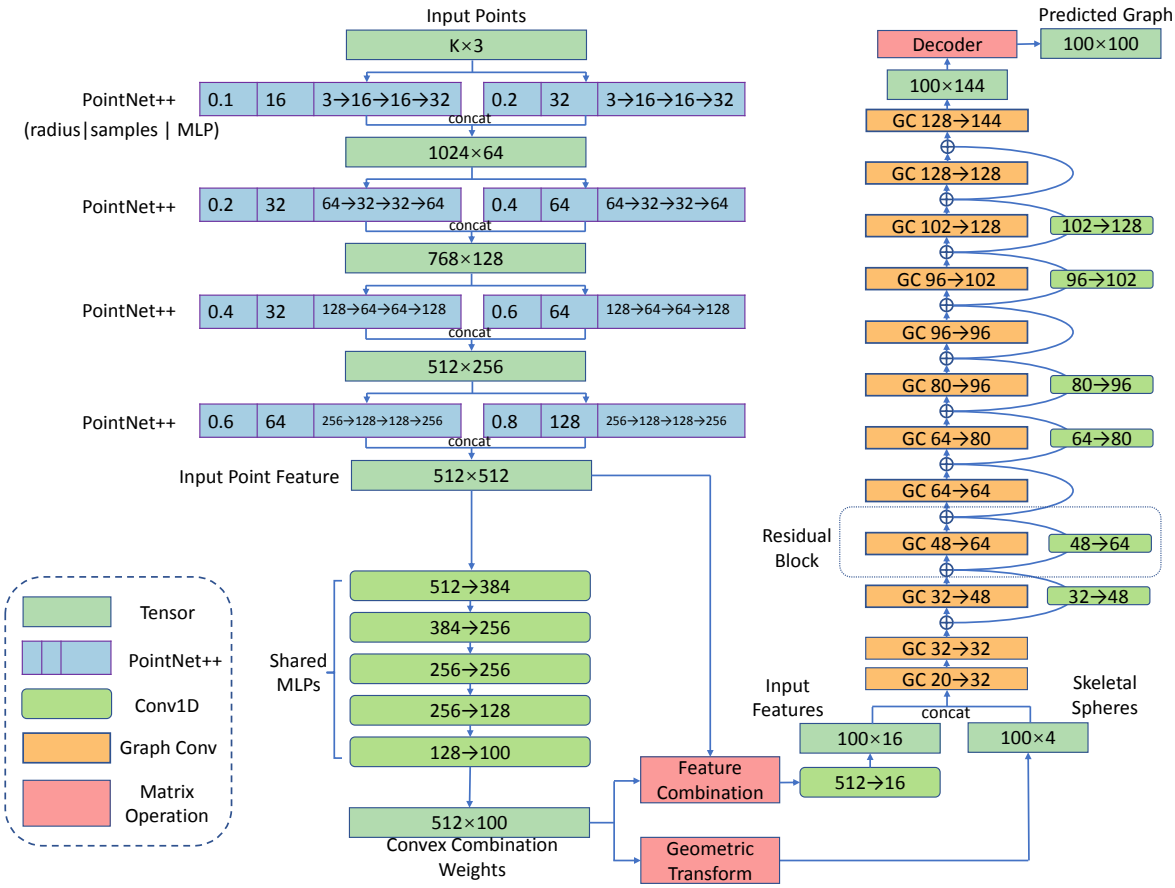

Figure 1. Overall network architecture of Point2Skeleton.

also followed by a batch normalization and a ReLU activation function. We use residual blocks [4] between the consecutive GCN layers, where the additional branching layers are used to align the dimension of features.

2. Rationale of Radius Computation

Given a set of densely sampled points on the boundary surface of a 3D shape, since the shape must be inside its convex hull, it is easy to show that any arbitrary point inside the shape can be derived from a convex combination of the sampled points, i.e., a linear combination with non-negative weights $\{w_i\}$ summing up to 1.

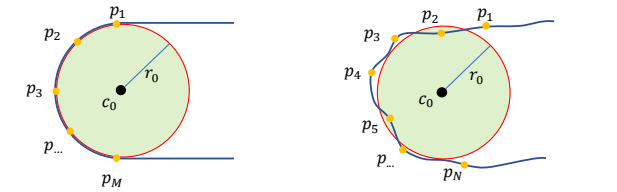

Figure 2. Illustration of the radius computation using closest distances. (a) An ideal case where (c_0, r_0) is a maximal inscribed sphere; (b) a typical case where the surface has noise and the radius is approximated by the closest distances.

Now we show why it is reasonable to use the same weights of convex combination to estimate the radius of a skeletal point based on their closest distances to the sampled points. Recall that, the closest distance from an input sample point p to all the skeleton points $\{c_i\}$ is defined as,

$$d(p, \{c_i\}) = \min_{c \in \{c_i\}} \|p - c\|_2 \,. \tag{1}$$

To simplify the discussion, we first analyze an ideal case. As shown in Fig. 2 (a), consider a skeletal sphere $s_0 = (c_0, r_0)$ that is maximally inscribed in a shape, where c_0 is the coordinate of the sphere center and r_0 the sphere radius. Assume the sphere has M ($M \geq 2$) touching points on the shape surface. Obviously, the M points are the closest points to c_0 on the shape surface and their distances to the center c_0 are all equal to the sphere radius. Thus we have

$$d(p_i, c_0) = r_0, \qquad \text{for} \quad i = 1, 2, ..., M. \tag{2}$$

Then, given an arbitrary group of convex combination weights $\{w_1, w_2, ..., w_M\}$ of the M points, if we use the weights to combine the closest distances, we always obtain a constant value which is the sphere radius:

$$\sum_{i=1}^{M} w_i d(p_i, c_0) = r_0, \quad \text{with} \quad \sum_{i=1}^{M} w_i = 1. \tag{3}$$

Figure 3. Visualization of the predicted combinational weights for the corresponding skeletal points. Each weight value w_i is scaled by w_i/w_{max} for better visualization, where w_{max} is the maximum weight value for the corresponding skeletal point.

We now analyze the typical case to generate a skeletal point. First, we visualize the learned combinational weights of the input points for certain skeletal points in Fig. 3. For a given skeletal point c, although the combinational weights to derive c are not unique, we observe that the weights are larger for the surface points that are very close to c, but are smaller or diminish to 0 for those far away from c. Therefore, the skeletal point c is approximated by the convex combination of a local set of input points that are closest to c. As shown in Fig. 2 (b), this is similar to the case discussed above. Hence, by using the same combinational weights, we approximate the radius by the weighted average of the closest distances, which therefore provides a reasonable estimation of the true radius at the skeletal point c.

3. Graph AutoEncoder

Given an undirected and unweighted graph with N nodes, the encoder is defined as a series of graph convolutional layers. Unlike most existing works that use a shallow GCN (usually no more than 4 layers [9]), we use a deep GCN with 12 layers to capture richer structures at various levels of abstraction. Consequently, to handle the degeneration problem caused by the depth of the network, we also include residual blocks [4] between consecutive layers. The encoder is given by:

$$\begin{gathered} GCN(\boldsymbol{X}^0, \boldsymbol{A}) = \tilde{\boldsymbol{A}}\boldsymbol{X}^{L-1}\boldsymbol{W}^{L-1}, \text{with} \\ \boldsymbol{X}^l = \sigma(\tilde{\boldsymbol{A}}\boldsymbol{X}^{l-1}\boldsymbol{W}^{l-1} + \boldsymbol{X}^{l-1}), \text{for } l \in \{1, ..., L-1\}. \end{gathered} \tag{4}$$

Here, $\boldsymbol{A} \in \{0,1\}^{N \times N}$ is the adjacency matrix, and $\tilde{\boldsymbol{A}}$ is the symmetrically normalized $\boldsymbol{A}$ given by $\tilde{\boldsymbol{A}} = \boldsymbol{D}^{-\frac{1}{2}}(\boldsymbol{A} + \boldsymbol{I}_N)\boldsymbol{D}^{-\frac{1}{2}}$, where $\boldsymbol{D}$ is the degree matrix of $\boldsymbol{A}$ and I_N the identity matrix indicating self-connections. $\boldsymbol{X}^0$ is the input node features and $\boldsymbol{X}^l$ the latent features. $\boldsymbol{W}^l$ is a layer-specific trainable weight matrix. σ is the ReLU activation function and L is the number of layers. The decoder we use is a simple inner product decoder to produce the reconstructed adjacency matrix $\hat{\boldsymbol{A}}$:

$$\hat{\boldsymbol{A}} = Sigmoid(\boldsymbol{Z}\boldsymbol{Z}^T) \quad \text{with} \quad \boldsymbol{Z} = GCN(\boldsymbol{X}^0, \boldsymbol{A}), \tag{5}$$

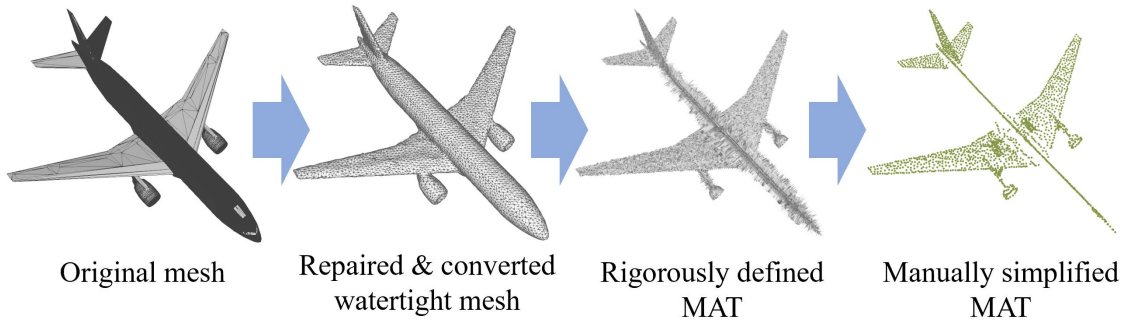

Figure 4. Steps to compute a simplified MAT from a surface mesh.

where $\boldsymbol{Z}$ is the learned latent features. By applying the inner product on the latent variables $\boldsymbol{Z}$ and $\boldsymbol{Z}^T$, we measure the similarity of each node inside $\boldsymbol{Z}$. The larger the inner product $z_i^T z_j$ in the embedding is, the stronger correlation the nodes i and j exhibit, which indicates that they are more likely to be connected.

We evaluate the effect of the number of the graph convolution layers. As shown in Fig. 5, a deeper GCN achieves better performance, i.e., smaller Masked Balanced Cross-Entropy (MBCE) loss [8].

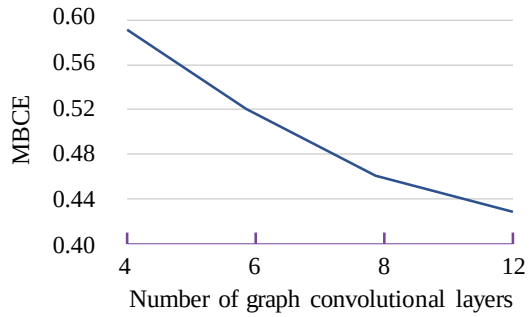

Figure 5. Evaluation of the number of graph convolutional layers.

Figure 6. More examples of the simplified MAT used for evaluation.

4. Standard MAT for Evaluation

Computation. As mentioned in the paper, there are no existing metrics to evaluate whether a skeletonization is reasonable. We use the manually simplified MAT that not only has meaningful structures but also exhibits good geometric accuracy, to evaluate different methods.

The main steps we adopt to generate a simplified MAT are shown in Fig. 4. For a point cloud in our dataset, we first find its ground-truth mesh in the ShapeNet [1]. Since the original mesh in the ShapeNet is not watertight either, we need to repair and convert the mesh to a strictly watertight one [5]. Then, we compute a standard MAT, which inevitably contains numerous insignificant spikes. After that, we manually simplify the MAT using a rule-based method [6]; we choose a certain threshold by which most spikes can be removed and the result is clean and structurally meaningful. Finally, we sample points on the simplified MAT for evaluating the CD and HD.

Fig. 6 shows more examples of the simplified MAT used for evaluation. It can be observed that the simplified MATs precisely capture the underlying structures of the input shapes using curve-like and surface-like components. Thus, they are suitable for evaluating whether the skeletonization of a method is reasonable and accurate.

These experiments, on the other hand, show that computing a clean and structurally meaningful skeletal representation by MAT simplification is difficult and expensive, given these tedious and time-consuming steps of geometric processing. By comparison, our method is more feasible, easier to use, and more efficient in practical applications.

Can we compute the MAT from the surface mesh reconstructed from a point cloud? Note that the simpli-

fied MATs in our dataset are computed by the ground-truth meshes corresponding to the point clouds. Considering that the surface mesh can also be reconstructed from a point cloud, one may wonder if we can directly reconstruct the surface meshes from the point clouds, and then compute the MATs based on the reconstructed surfaces. Through the experiments, we find this strategy is not feasible, since the surface quality of the existing surface reconstruction methods (like Poisson reconstruction) cannot satisfy the requirement of MAT computation. As a result, the MAT computation algorithm crashes in most cases.

5. Effect of Input Quality

We show more detailed quantitative results for evaluating the effect of the input quality to our method. As shown in Table 1, we input point clouds with different point numbers and noise levels, i.e., 2000 points without noise, 1000 points with 0.5% noise, and 500 points with 1% noise, to our method; the quantitative results are largely similar. Besides, we test some point clouds with uneven density distribution, of which results are given in Fig. 7; our method is robust to the density variation of point cloud.

	CD-Recon	HD-Recon	CD-MAT	HD-MAT
2000 - 0%	0.0372	0.1424	0.0828	0.1898
1000 - 0.5%	0.0382	0.1615	0.0851	0.2071
500 - 1%	0.0458	0.2127	0.0958	0.2524

Table 1. Quantitative evaluation results on different number of input points and different levels of noise.

Figure 7. Generated skeletal meshes for point clouds with uneven density distribution.

6. Effect of Skeletal Point Number

To study how the number of skeletal points (N) affects the results of skeletal mesh generation, we use $N = 200, 100$ and 50 for evaluation. The qualitative and quantitative results are given in Fig. 8 and Table 2. The results suggest that, with the other conditions unchanged, using more skeletal points leads to lower reconstruction error but includes more insignificant details, which reduces the simplicity and abstraction level. To balance the accuracy and structural simplicity, we use $N = 100$ in the paper.

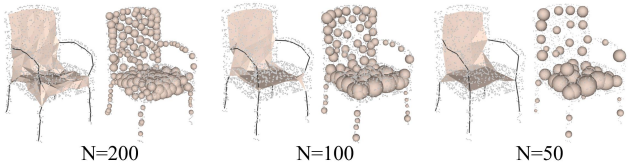

Figure 8. Results of skeletal mesh prediction using different number of skeletal points (N).

	CD-Recon	HD-Recon	CD-MAT	HD-MAT
N = 200	0.0310	0.1402	0.0782	0.2002
N = 100	0.0372	0.1424	0.0828	0.1898
N = 50	0.0494	0.1720	0.0990	0.2225

Table 2. Quantitative evaluation results on different number of skeletal points (N).

7. Limitations and Failure Cases

For the skeletal point prediction, as mentioned in the paper, we use convex combination of the input points to generate the skeletal points. Therefore, our method will fail to recover a partial point cloud if its original skeleton cannot be completely included in the convex hull of the partial shape.

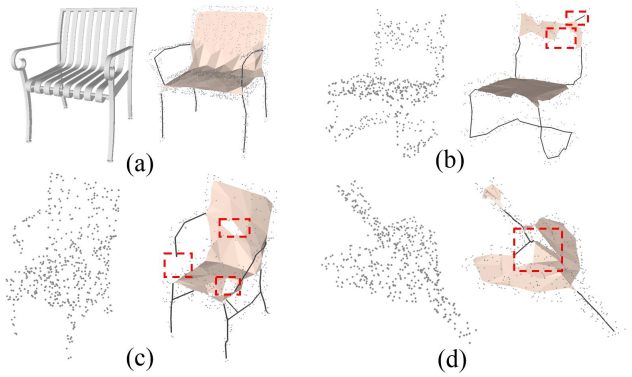

Figure 9. Failure cases of the skeletal mesh generation.

Generally, we observe the prediction of skeletal points is stable driven by the shape geometry, while connecting the skeletal points to faithfully capture the shape structure is more challenging. We show some failure cases of the mesh generation in Fig. 9. First, we use local connectivity as a prior to initialize the edge connections of the graph. However, the local connectivity is not always true but based on the assumption that the point cloud has a relatively distinguishable structure. As shown in Fig. 9 (a), solely using

local connectivity cannot enforce all the connections to be located inside the shape, leading to inconsistent structures with the original shape. Giving additional inside/outside labels as supervision like [2, 3] would be a potential solution.

Second, although we use some strategies to make the mesh generation more reliable, some results still have unsmooth or incorrect structures, especially when the point cloud is noisy and sparse, as shown in Fig. 9 (b)(c)(d). On the one hand, without ground truth data, this is an inherent challenge for the unsupervised learning to exactly capture the detailed shape geometry. On the other hand, the link prediction of GAE is only based on the correlations of the skeletal points in the latent space, of which learned features remain not fully explained. Incorporating explicit topological constraints in the network would help improve the mesh quality.

References

[1] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 3

[2] Zhiqin Chen and Hao Zhang. Learning implicit fields for generative shape modeling. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5939–5948, 2019. 5

[3] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7154–7164, 2019. 5

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 2

[5] Jingwei Huang, Hao Su, and Leonidas Guibas. Robust watertight manifold surface generation method for shapenet models. *arXiv preprint arXiv:1802.01698*, 2018. 3

[6] Pan Li, Bin Wang, Feng Sun, Xiaohu Guo, Caiming Zhang, and Wenping Wang. Q-mat: Computing medial axis transform by quadratic error minimization. *ACM Transactions on Graphics (TOG)*, 35(1):1–16, 2015. 3

[7] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in neural information processing systems*, pages 5099–5108, 2017. 1

[8] Phi Vu Tran. Learning to make predictions on graphs with autoencoders. In *2018 IEEE 5th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 237–245. IEEE, 2018. 3

[9] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020. 2