HPNet: Deep Primitive Segmentation Using Hybrid Representations – Supplementary Material

A. Additional Analysis of the Spectral Embedding Module

This section provides a proof on the following result of the main paper, which states that when $n_k = \frac{n}{K}$,

$$\min_{R\in O(K)} \frac{\|U_c R - U_c^g\|_{\mathcal{F}}}{\|U_c^g\|_{\mathcal{F}}} = O\left(\frac{2K\sqrt{\rho}}{(1-\rho)+\sqrt{1-\rho}} n^{-\frac{1}{4}}\right). \quad (1)$$

To obtain this result, we first look at E. It is clear that if the prediction associated with a point is incorrect, then it would affect $O(\sqrt{n})$ edges. Therefore,

$$\|E\|_{\mathcal{F}} \in O\left(\sqrt{\rho} n^{\frac{3}{4}}\right). \quad (2)$$

Likewise, if the shape prediction associated with a point is correct, it will introduce edges for each point within the same primitive. Suppose the size of each primitive is n_k and the number of correct predictions $(1-\rho)n_k$. We proceed to characterize the spectrum of $A_{c,k}^g$.

Proposition 1 *The eigenvalues of $A_{c,k}^g$ are given by*

$$\lambda_i(A_{c,k}^g) = \begin{cases} n_k \frac{(1-\rho)+\sqrt{1-\rho}}{2}, & i = 1 \\ 0, & 1 < i < n_k \\ n_k \frac{(1-\rho)-\sqrt{1-\rho}}{2}, & i = n_k. \end{cases} \quad (3)$$

Proof: Proof of Section A.1.

Suppose $n_k = \frac{n}{K}$. We can derive (1) by substituting (2) and (3) into the following variant of the Davis-Kahan theorem which is copied from the main paper:

$$\min_{R\in O(K)} \|U_c R - U_c^g\|_{\mathcal{F}} \leq \frac{\sqrt{\lambda_1(A_c^g)}\|E\|_{\mathcal{F}}}{\lambda_K(A_c^g) - \lambda_{K+1}(A_c^g)}. \quad (4)$$

(1) implies when the fractions of wrong shape predictions are small, the spectral gap $\lambda_K(A_{c,k}^g) - \lambda_{K+1}(A_{c,k}^g)$ is large, which implies the advantage of using U_c^g over the predicted shape parameters for clustering.

A.1. Proof of Prop. 1

Without losing generality, we assume the correct predictions occupy the first ρn_k elements. It it easy see to check that

$$A_{c,k}^g = \begin{pmatrix} \mathbf{1} \\ \frac{1}{2}\mathbf{1} \end{pmatrix} \begin{pmatrix} \mathbf{1} \\ \frac{1}{2}\mathbf{1} \end{pmatrix}^T - \frac{1}{4} \begin{pmatrix} \mathbf{0} \\ \mathbf{1} \end{pmatrix} \begin{pmatrix} \mathbf{0} \\ \mathbf{1} \end{pmatrix}^T.$$

Therefore, the rank of $A_{c,k}^g$ is 2. It is easy to check that the two non-zero eigenvalues of $A_{c,k}^g$ are

$$\lambda_1(A_{c,k}^g) = n_k \frac{(1-\rho)+\sqrt{(1-\rho)}}{2}$$
$$\lambda_{n_k}(A_{c,k}^g) = n_k \frac{(1-\rho)-\sqrt{1-\rho}}{2},$$

and the corresponding eigenvectors are

$$\boldsymbol{n}_1(A_{c,k}^g) = \begin{pmatrix} (1+\frac{1}{\sqrt{1-\rho}})\mathbf{1} \\ \mathbf{1} \end{pmatrix},$$
$$\boldsymbol{n}_{n_k}(A_{c,k}^g) = \begin{pmatrix} (1-\frac{1}{\sqrt{1-\rho}})\mathbf{1} \\ \mathbf{1} \end{pmatrix}.$$

which ends the proof.

B. Geometric Consistency Matrix Details

Here, we describe our approach for computing the distance metric $d(p_i, \boldsymbol{s}_j)$ efficiently.

Firstly, Dense Descriptor Module predicts three dense point-wise attributes, including a semantic feature descriptor $\boldsymbol{d} \in \mathbb{R}^{N\times 128}$, a binary type indicator vector $\boldsymbol{t} \in \{0,1\}^{N\times 6}$, and a shape parameter vector $\boldsymbol{s} \in \mathbb{R}^{N\times 22}$, where N is the number of points. Then we process each primitive type t respectively. If primitive type $t \in \{\text{Plane}, \text{Sphere}, \text{Cone}, \text{Cylinder}\}$, we extract the points belonging to this type according to the binary type indicator. Let M be the number of points from this primitive type. Take Plane as an example, we can get the corresponding shape parameter vector $s \in \mathbb{R}^{M\times 4}$. If primitive type $t \in \{\text{B-spline-Open}, \text{B-spline-Closed}\}$, we first extract the points and take them as input to the SplineNet [2] to get the control points of B-spline patch. Then we can compute a sub-distance matrix $d' \in \mathbb{R}^{N\times M}$, where $d'_{ij} = d(p_i, \boldsymbol{s}_j)$. Therefore, for each shape, we only need 6 iterations for each primitive type which is feasible in practice.

When computing the distance between a point and a Bspline patch, we first randomly sample each Bspline patch (1024 points), the samples are reused for all the other points. For nearest neighbor query, we used a grid data structure.

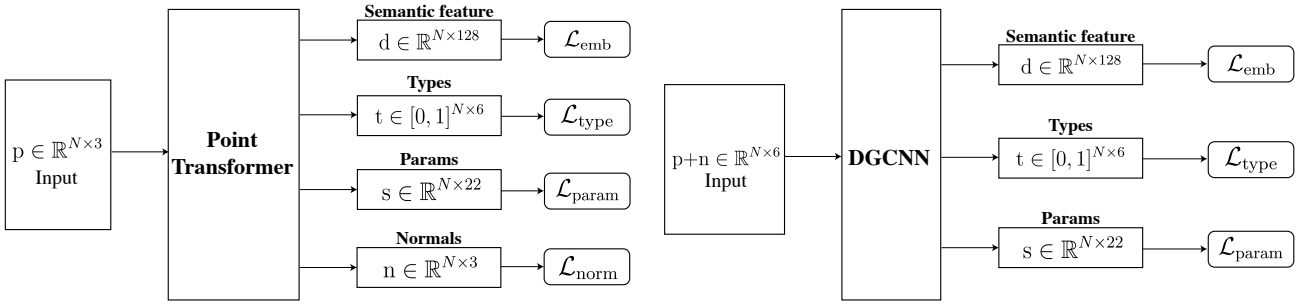

Figure 1: Network architecture of Dense Descriptor Module. (a). With only point position as input. (b). With point position and normal as input.

C. Architecture Details of Dense Descriptor Module

Dense Descriptor Module takes a point cloud $\mathcal{P} = \{p_i | 1 \leq i \leq n\}$ as input. Each point p_i has a position $\boldsymbol{p}_i \in \mathbb{R}^3$ and an optional normal $\boldsymbol{n}_i \in \mathbb{R}^3$. As illustrated in Figure 1(a), with only point position as input, The dense descriptor module uses Point Transformer [4] as backbone, and outputs four dense point-wise attributes. The attributes associated with each point $p_i \in \mathcal{P}$ includes a semantic feature descriptor $\boldsymbol{d}_i \in \mathbb{R}^{128}$, a binary type indicator vector $\boldsymbol{t}_i \in \{0, 1\}^6$, a shape parameter vector $\boldsymbol{s}_i \in \mathbb{R}^{22}$, and the point normal $\boldsymbol{n}_i \in \mathbb{R}^3$.

As illustrated in Figure 1(b), when adding normal as input, the dense descriptor module replaces point transformer with DGCNN [3]. In our experiments, we found that it has a strong ability to capture feature information of surface normal. Note that this module outputs three dense point-wise attributes.

D. Training Details

Dataset. The ANSI Mechanical Component Dataset [1] is provided by TraceParts. Four basic primitive types (plane, sphere, cylinder, cone) cover 94% percentage area per-model on average in ANSI. The maximum number of primitives per shape does not exceed 20 in all the models. In experiments, we first uniformly sample 8192 points over the entire surface of each shape as the input point cloud . Then we normalize each shape so that its mean is at the origin, and the diameter of the shape is 1. During training, we perturb the input model by randomly moving each point along along the surface normal direction with a random value in $[-0.01, 0.01]$.

The ABCParts dataset is derived from the ABC dataset [2]. Each model consists of at least one B-spline patch. In the experiments, we first sample each shape with 10K points randomly distributed on the shape surface as the input point cloud. Similar to ANSI, during training, we also randomly perturb each data point along the surface normal direction in $[-0.01, 0.01]$ range. The normals are also perturbed with random noise in a uniform range of $[-3, 3]$ degrees from their original direction. All the baseline approaches share the same data preprocessing procedure with our method.

	Input	Pointnet++	DGCNN	PointTransformer
SPFN	p	47.38	53.83	58.15
SPFN	p+n	69.01	73.41	70.14
ParseNet	p	63.14	71.32	75.01
ParseNet	p+n	73.53	82.14	76.13
Ours	p	71.45	74.99	78.12
Ours	p+n	76.10	85.24	80.22

Table 1: Segmentation mean IoU results of our method and baseline approaches with different backbones on ABCParts Dataset.

Training Procedure Details. Network training of HPNet consists of two stages. The first stage learns the Dense Descriptor Module. Without normal as input, we use default hyperparameters of network with batch size 32, initial learning rate 1×10^{-2}. Learning rate reduces by the factor of 10 when the validation performance convergence. The experiment took 20 hours on a Tesla V100 GPU. With normal as input, we use default hyperparameters of network with batch size 8, initial learning rate 1×10^{-3}. Learning rate reduces by the factor of 2 when the validation performance converges. The experiment on each dataset took 50 hours on a Tesla V100 GPU.

Baseline Comparison. Previous baseline approaches used different network backbones compared with our method. To make fair comparison, we report the experimental results under different backbones in Table 1. The results demonstrate that our approach leads to noticeable performance improvements over baseline methods (the absolute improvements vary under different settings slightly). Note that replacing PointNet++ in SPFN by DGCNN does not violate that ParseNet is the top-performing baseline.

E. Visualization of Different Feature Combinations

Here, we show more visualization results to continue to study the impacts of different components of HPNet.

Figure 2 shows qualitative results for comparisons between our approach without the consistency spectral

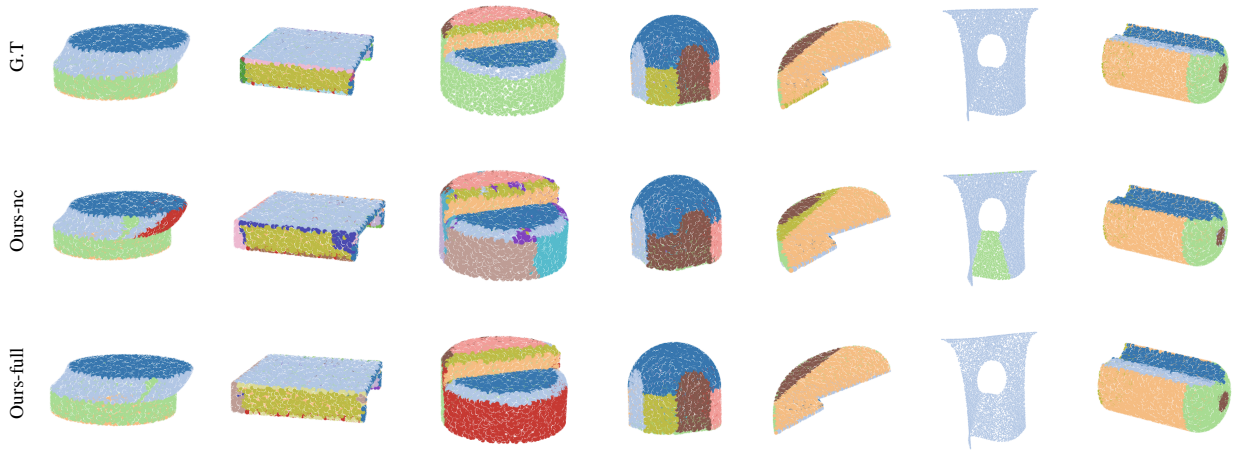

Figure 2: Comparison between our approach without combining the consistency spectral descriptors('Ours-nc') and our full approach ('Ours-full').

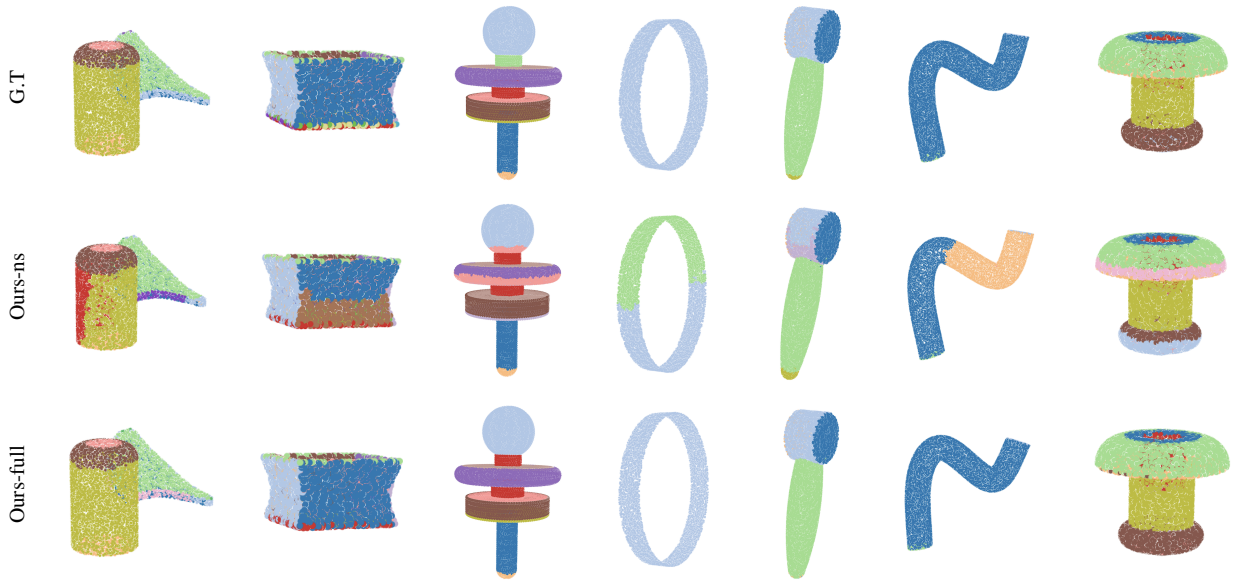

Figure 3: Comparison between our approach without combining the smoothness spectral descriptors('Ours-ns') and our full approach ('Ours-full').

descriptors and our full approach. When the model shows accurate predictions of primitive parameters, combining geometry consistency spectral descriptor can segment the primitive patch more precisely (columns 2 and 4). This is because of the points from the same primitive share nearly the same primitive parameters. In this case, the geometry consistency descriptor can also help merge two patches that belong to the same primitive into one patch (columns 1, 3, and 6). Moreover, when two different primitive patches are erroneously merged, it can help separate them (columns 4 and 7).

Figure 3 shows more results that compare our approach without the smoothness spectral descriptors with our full approach. Besides capturing sharp edges between different primitives, the smoothness spectral descriptor can also help rectify two patches when their boundaries are smooth.

References

[1] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, Li Yi, and Leonidas J Guibas. Supervised fitting of geometric primitives to 3d point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2652–2660, 2019. 2

[2] Gopal Sharma, Difan Liu, Subhransu Maji, Evangelos Kalogerakis, Siddhartha Chaudhuri, and Radomír Mech. Parsenet: A parametric surface fitting network for 3d point clouds. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings,*

Part VII, volume 12352 of *Lecture Notes in Computer Science*, pages 261–276. Springer, 2020. 1, 2

[3] Yue Wang, Yongbin Sun, Ziwei Liu, Sanjay E Sarma, Michael M Bronstein, and Justin M Solomon. Dynamic graph cnn for learning on point clouds. *Acm Transactions On Graphics (tog)*, 38(5):1–12, 2019. 2

[4] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer, 2020. 2