HPNet: Deep Primitive Segmentation Using Hybrid Representations

Siming Yan\textsuperscript{1} Zhenpei Yang\textsuperscript{1} Chongyang Ma\textsuperscript{2} Haibin Huang\textsuperscript{2}
Etienne Vouga\textsuperscript{1} Qixing Huang\textsuperscript{1}
\textsuperscript{1}The University of Texas at Austin \textsuperscript{2}Kuaishou Technology

Abstract

This paper introduces HPNet, a novel deep-learning approach for segmenting a 3D shape represented as a point cloud into primitive patches. The key to deep primitive segmentation is learning a feature representation that can separate points of different primitives. Unlike utilizing a single feature representation, HPNet leverages hybrid representations that combine one learned semantic descriptor, two spectral descriptors derived from predicted geometric parameters, as well as an adjacency matrix that encodes sharp edges. Moreover, instead of merely concatenating the descriptors, HPNet optimally combines hybrid representations by learning combination weights. This weighting module builds on the entropy of input features. The output primitive segmentation is obtained from a mean-shift clustering module. Experimental results on benchmark datasets ANSI and ABCParts show that HPNet leads to significant performance gains from baseline approaches.

1. Introduction

The geometry of man-made objects can be frequently analysed in terms of primitive surface patches (planes, spheres, cylinders, cones, and other simple parametric surfaces). Decomposing a 3D model into primitive surfaces is of fundamental importance with applications in reverse engineering, shape compression, shape understanding, shape editing, and robot learning. Primitive segmentation is the task of grouping and labeling points on an object based on primitive shape, and is fundamentally challenging due to the large search space and the fact that primitive patches may only approximately fit the object.

This paper introduces a deep learning model called HPNet. It takes as input a point cloud (optionally including normals) and outputs a segmentation of the point cloud into primitives, with a type label for each primitive segment (see Figure 1). The main idea of HPNet is to combine traditional tried-and-true geometric heuristics for primitive detection (for instance, algebraic relations between points and shape primitives, and segmentation from sharp edges) with a deep primitive detection approach based on powerful feature learning. We achieve this union through the use of a hybrid point descriptor that combines a learned semantic descriptor and two spectral descriptors. The first spectral descriptor is derived from the adjacency matrix between the input points and predicted primitive parameters. The second spectral descriptor is built based on the adjacency matrix that models sharp edges. In both cases, the spectral approach unifies all primitive segmentation cues as point descriptors. It also rectifies the inputs to the spectral modules, e.g., incorrect predictions of primitive parameters. Given the point descriptors, HPNet employs a mean-shift clustering module to perform primitive segmentation. We also present an effective approach to train HPNet.

A key insight that allows HPNet to succeed on a wide diversity of input shapes is that different types of features have different power on different types of models. For example, when planar shapes are prominent, spectral descriptors are very effective, as eigenvectors of edge-aware adjacency operators localize strongly on the planar patches and segment them robustly. In contrast, the semantic descriptors are more useful for complex curved shapes, such as patches of cones and cylinders, in which local curvature information is sufficient to determine the shape parameters. HPNet exploits this insight by using adaptive weights to combine different types of features. A weighting module automatically computes the relative weighting for different types of descriptors, based on an entropy measure of each descriptor which estimates the importance of that descriptor for clustering.

We evaluate HPNet on two benchmark datasets, i.e., ANSI [14] and ABCParts [23]. With only points as input,
HPNet improves the Segmentation Mean IoU (MIoU) score from 80.9/75.0 to 91.3/78.1 on the ANSI and ABCParts benchmarks, respectively. With both positions and normals as input, HPNet improves the MIoU score from 88.6/82.1 to 94.2/85.2. We also perform an ablation study to justify the effectiveness of different components of HPNet. Our code is available at https://github.com/SimingYan/HPNet.

2. Related Work

Primitive segmentation from 3D shapes has been studied considerably in the past. It is beyond the scope of this paper to provide a comprehensive overview. We refer to [8] for a thorough survey.

Non-deep learning based approaches. Traditional approaches leverage stochastic paradigms [5, 21, 2], parameter spaces [20] or clustering and segmentation techniques [31, 12, 7]. Among these early works, RANSAC [5] and its variants [21, 15, 17, 9] are the most widely used methods for primitive segmentation and fitting. RANSAC-based methods can estimate model parameters iteratively and showed state-of-the-art results. However, they also suffer from laborious parameter tuning for each primitive. They also do not fully utilize each shape’s information (e.g., sharp edges) and prior knowledge about primitive segments, which can be incorporated via machine learning approaches.

Deep learning based approaches. Several recent works [23, 29, 39, 14, 25] have studied how to develop deep learning models for primitive segmentation. [39] and [29] proposed to detect cuboids as rough abstractions of the input shapes. However, their performance is limited on other types of primitives. CSGNet [24] can handle more variety of primitives, but it requires a labeled hierarchical structure for the underlying primitives. This hierarchical structure is not always well-defined.

Our work is most relevant to SPFN [14] and ParseNet [23]. SPFN proposed a supervised primitive fitting method to predict per-point properties, including segmentation labels, type labels, and normals. Then they introduced a differential model estimation module to fit the primitive parameters. ParseNet proposed a complete model that can handle more primitives, including B-spline patches. However, in the segmentation part of their work, they mainly utilized semantic supervision, which ignored the importance of combining geometric features such as sharp edges. Another novelty of HPNet is that we design the prediction of per-point shape parameters and leverage spectral embedding to generate clearer dense point-wise descriptors.

Hybrid representations for 3D recognition. Our approach is motivated by recent methods that use hybrid geometric representations for solving 3D vision tasks. Examples include leveraging different types of keypoints for relative pose estimation [6, 32], utilizing hybrid geometric primitives for 3D object detection and segmentation [35, 36], and geometric synthesis under hybrid representations [37, 19]. Our work differentiates from prior methods by learning two spectral descriptors and a weighting sub-module that combines different geometric representations. In particular, the weighting sub-module models entropy for feature representations. This functionality is hard to achieve using alternative techniques, e.g., feature transformation networks. The weighting sub-module is also relevant to the line of work on feature selection [28, 16, 13]. The resulting weights are derived from solving a quadratic program.

3. Overview

3.1. Problem Statement

We assume the input is given by a point cloud \( \mathcal{P} = \{p_i\}_{1 \leq i \leq n} \). Each point \( p_i \) has a position \( \mathbf{p}_i \in \mathbb{R}^3 \) and an optional normal \( \mathbf{n}_i \in \mathbb{R}^3 \). Our goal is to decompose \( \mathcal{P} = \mathcal{P}_1 \cup \cdots \cup \mathcal{P}_K \) into \( K \) primitives \( \mathcal{P}_k, 1 \leq k \leq K \). Each primitive has a type (i.e., plane, sphere, cylinder, cone, b-spline) and some type-specific shape parameters (normal; radius; control points; etc). We translate and scale each point cloud so that its mean is at the origin, and the diameter of the point cloud is 1.

3.2. Overview of HPNet

Figure 2 illustrates the pipeline of HPNet, which combines three types of modules: dense descriptor, spectral embedding, and clustering.

Dense descriptor module. The first module is trained to compute dense point-wise features. These features consist of a semantic descriptor trained to differentiate points from different primitives, a binary type vector identifying the primitive type each point likely belongs to, a parameter vector that predicts the shape parameters of the primitive fitting the point’s neighborhood, and the point normal (in case normals were not provided in the input). Note that unlike standard approaches that either learn descriptors to separate different primitive points or detect shape parameters (c.f. [8, 14, 25]), HPNet combines both approaches. Incorporating both types of features promotes HPNet learning both semantic information (encoded in the type vectors) and local shape geometry (encoded in the shape parameters).

Spectral embedding modules. The spectral embedding module converts relational cues useful for segmentation (algebraic relations between points that indicate consistency in surface shape parameters, or presence of sharp edges) into dense point-wise descriptors. Each type of relational cue is modeled using an adjacency graph with different weights. The resulting point descriptors are given by the leading eigenvectors of this adjacent matrix. We consider two relational cues that are complementary to the semantic point descriptors. The first module identifies sharp edges,
which serve as boundaries between many pairs of primitive patches. Its adjacency matrix assigns high weights to neighboring vertices with similar normals. As the Euclidean distance in the spectral embedding space corresponds to the diffusion distance on the adjacency graph, points that fall on different sides of a sharp edge are far from each other in the embedding space, allowing robust identification of patches separated by creases.

The second module models the consistency between the input points and the predicted shape parameters. It assigns high weight to an edge if one endpoint is consistent with the local geometry predicted by the other endpoint’s shape parameters. The advantage of this formulation comes from the stability of leading eigenvectors against matrix perturbations. Even when a significant part of the predicted primitive parameters is incorrect, the number of edges with wrong weights due to wrong primitive parameters is typically small, so that the correct predictions still yield an adjacency matrix where each primitive patch is a strongly connected sub-graph. The resulting descriptors are usually cleaner than the raw predicted shape parameters from the dense descriptor module.

**Clustering module.** The clustering module aggregates the point descriptors and applies mean-shift clustering to obtain the final primitive decomposition. A key observation is that the desired combination weights vary across different models. Instead of relying on hand-crafted weights, HPNet learns the combination weights by building on an entropy metric that assesses the underlying clustering structures in high-dimensional point clouds.

**Network training.** We present an effective strategy to train HPNet. The main idea, which has proven to be successful for keypoint-based 6D object pose estimation [18], is to use a large-scale training set and a small-scale validation set. The training set is used in learning the dense descriptor module. In contrast, the validation set is used in learning the hybrid parameters of the spectral embedding and clustering modules. This approach alleviates over-fitting.

### 4. Our Method

This section introduces our approach in detail. Section 4.1 to Section 4.3 present the three modules of HPNet. Section 4.4 then introduces how to train HPNet in an end-to-end manner.

#### 4.1. Dense Descriptor Module

As shown in Figure 2, the first module of HPNet predicts dense point-wise attributes. The attributes associated with each point \( p_i \in \mathcal{P} \) includes a semantic feature descriptor \( d_i \in \mathbb{R}^{128} \), a binary type indicator vector \( t_i \in \{0, 1\}^{6} \), and a shape parameter vector \( s_i \in \mathbb{R}^{22} \). HPNet considers six primitive types: Plane, Sphere, Cylinder, Cone, B-spline-Open, and B-spline-Closed. The type indicator vector satisfies \( t_i(j) = 1 \) if and only if the index of the underlying primitive type of \( p_i \) is \( j \). \( s_i \) collects shape parameters for the Plane, Sphere, Cylinder, and Cone. HPNet uses the pre-trained SplineNet introduced in [25] to get the control points of open and closed b-spline patches. The prediction network is derived from DGCNN [30] and Point Transformer [38], and the details are deferred to the supplemental material.

#### 4.2. Spectral Embedding Modules

The spectral embedding modules take adjacency matrices of the input point cloud as inputs and output their leading eigenvectors. Each leading eigenvector is then a dense point descriptor function. HPNet considers two adjacency matrices. The first one models the consistency between the positions of the input points and the predicted shape parameters in a neighborhood around the point. The second one models presence of sharp edges. We provide the details for constructing these matrices in Sections 4.2.1 and 4.2.2 below.
4.2.1 Geometric Consistency Matrix

We first define a distance metric \( d(p_i, s_j) \) between one point \( p_i \) and the predicted shape parameter \( s_j \) associated with another point \( p_j \) [14]. Let \( t_j \) be the predicted primitive type of \( s_j \). When \( t_j \in \{ \text{Plane, Sphere, Cone, Cylinder} \} \), we define \( d(p_i, s_j) = \)
\[
\begin{cases}
\| p_i^T n_j - d_j \| & t_j = \text{Plane} \\
\| p_i - o_j \| - r_j & t_j = \text{Sphere} \\
\| (I - a_j a_j^T) (p_i - o_j) \| - r_j & t_j = \text{Cylinder} \\
\| p_i - o_j \| \cos \left( \arccos \left( \frac{p_i^T a_j}{\| p_i \| \| a_j \|} \right) - \theta_j \right) & t_j = \text{Cone}
\end{cases}
\]
where \( n_j \) and \( d_j \) denote the normal and distance of a plane primitive; \( o_j \) and \( r_j \) the center and radius of a sphere primitive; \( a_j, o_j, \) and \( r_j \) the direction, center, and radius of a cylinder; and \( o_j, a_j, \) and \( \theta_j \) the center, apex, and angle of a cone. There are 22 total parameters.

When \( t_j \in \{ \text{B-spline-Open, B-spline-Closed} \} \), we define \( d(p_i, s_j) \) as the closest distance between \( p_i \) and the B-spline patch specified by \( s_j \). Please refer to the supplemental material for an efficient approach for computing \( d(p_i, s_j) \) approximately. We then define the signed weight between \( p_i \) and \( s_j \) as
\[
w(p_i, s_j) := \exp \left( -\frac{d^2(p_i, s_j)}{2\sigma_{t_j}^2} \right),
\]
where \( \sigma_{t_j} > 0 \) is a hyperparameter associated with type \( t_j \). Finally, we define the consistency adjacency matrix \( A_c \in \mathbb{R}^{n \times n} \), whose elements are given by
\[
A_c(i, j) = \left( w(p_i, s_j) + w(p_j, s_i) \right) / 2, \quad 1 \leq i, j \leq n.
\]

Let \( \lambda_{c,i} \) and \( \mathbf{u}_{c,i} \) denote the \( i \)-th eigenvalues and eigenvectors of \( A_c \). We define the resulting descriptors as columns of \( U_c \in \left( \sqrt{\frac{\lambda_1}{\lambda_{c,1}}} \mathbf{u}_{c,1}, \ldots, \sqrt{\frac{\lambda_n}{\lambda_{c,n}}} \mathbf{u}_{c,n} \right) \in \mathbb{R}^{n \times d_c} \). Depending on the number of primitives, \( d_c \) varies across different shapes in our experiments.

4.3. Clustering Module

The clustering module concatenates the different types of point descriptors defined above, namely, the semantic descriptors \( \mathbf{d}_c \) and the two spectral features specified by \( U_c^T \mathbf{e}_i \) and \( U_c^T \mathbf{e}_j \). As mentioned in the introduction, the relative importance of the different descriptors varies dramatically based on the object geometry; so rather than using fixed weights to combine the descriptors, we introduce an approach that learns how to combine them.
We model the weight of each feature where

\begin{equation}
L = \sum_{i} w_i \leq 1,
\end{equation}

where there are \(m_i\) the dimension of the \(l\)-th feature. Our goal is to compute a weight \(w_i \in (0, 1)\) for each feature, with \(\sum_i w_i = 1\). In the context of this paper, \(L = 1 + d_1 + d_2\), i.e., \(F_1\) corresponds to the semantic descriptors \(d_1\), and each remaining \(F_l\) corresponds to one spectral descriptor. Note that we weight each spectral descriptor individually because the desired number of spectral descriptors is dependent on the number of underlying primitives, which varies across different shapes.

HPNet uses the criterion that a feature \(F_l\) should have large weights if \(F_l\) reveals articulated cluster structure. Motivated from the feature selection for clustering approach described in [4], HPNet applies an entropy score to define the feature weight. Specifically, we first model the multivariate probability function of the feature space \(F_l\) as

\begin{equation}
P^\sigma_{F_l}(x) := \frac{1}{n} \frac{1}{(2\pi)^{m_l} \sigma_l^{m_l}} \exp \left(-\frac{\|x - F_l^T e_i\|^2}{2\sigma_l^2}\right),
\end{equation}

where \(\sigma_l\) is a hyperparameter associated with \(F_l\). The entropy of \(F_l\) is then

\begin{equation}
H_{\sigma_l}(F_l) := -\sum_{i=1}^n P^\sigma_{F_l}(F_l^T e_i) \log \left(P^\sigma_{F_l}(F_l^T e_i)\right).
\end{equation}

Intuitively, features where points form clusters (versus random point distributions) tend to have low entropy values. We model the weight of each feature \(F_l\) so that it is inversely proportional to \(H_{\sigma_l}(F_l)\):

\begin{equation}
w_l := \frac{1}{\sqrt{\sum_i w_i^2}}, \quad \overline{w}_l := \frac{1}{H_{\sigma_l}(F_l)}. \tag{4}
\end{equation}

### Weighting sub-module

To make the notations uncluttered, we describe our weighting scheme in a generic settings where there are \(L\) features \(F_l \in \mathbb{R}^{n \times m_l}, 1 \leq l \leq L\), with \(m_l\) the dimension of the \(l\)-th feature. Our goal is to compute a weight \(w_i \in (0, 1)\) for each feature, with \(\sum_i w_i = 1\). In the context of this paper, \(L = 1 + d_1 + d_2\), i.e., \(F_1\) corresponds to the semantic descriptors \(d_1\), and each remaining \(F_l\) corresponds to one spectral descriptor. Note that we weight each spectral descriptor individually because the desired number of spectral descriptors is dependent on the number of underlying primitives, which varies across different shapes.

HPNet uses the criterion that a feature \(F_l\) should have large weights if \(F_l\) reveals articulated cluster structure. Motivated from the feature selection for clustering approach described in [4], HPNet applies an entropy score to define the feature weight. Specifically, we first model the multivariate probability function of the feature space \(F_l\) as

\begin{equation}
P^\sigma_{F_l}(x) := \frac{1}{n} \frac{1}{(2\pi)^{m_l} \sigma_l^{m_l}} \exp \left(-\frac{\|x - F_l^T e_i\|^2}{2\sigma_l^2}\right),
\end{equation}

where \(\sigma_l\) is a hyperparameter associated with \(F_l\). The entropy of \(F_l\) is then

\begin{equation}
H_{\sigma_l}(F_l) := -\sum_{i=1}^n P^\sigma_{F_l}(F_l^T e_i) \log \left(P^\sigma_{F_l}(F_l^T e_i)\right).
\end{equation}

Intuitively, features where points form clusters (versus random point distributions) tend to have low entropy values. We model the weight of each feature \(F_l\) so that it is inversely proportional to \(H_{\sigma_l}(F_l)\):

\begin{equation}
w_l := \frac{1}{\sqrt{\sum_i w_i^2}}, \quad \overline{w}_l := \frac{1}{H_{\sigma_l}(F_l)}. \tag{4}
\end{equation}

### Mean-shift clustering sub-module

Since the primitive number varies between different models, we apply a mean-shift clustering procedure [3] to obtain the primitive segmentation result.

#### 4.4. Network Training

The network training of HPNet consists of two stages. The prediction module is trained in the first stage, while the hyperparameters of HPNet is learned in the second stage.

### 4.4.1 Training of the Prediction Module

We train the prediction module using a training dataset, in which each point cloud has ground-truth primitive segmentation and associated primitive parameters. In the following, we focus on defining the loss for one point cloud \(P\). The total loss consists of three terms:

\begin{equation}
L(P) = L_{emb}(P) + \alpha L_{type}(P) + \beta L_{param}(P), \tag{5}
\end{equation}

where \(L_{emb}(P)\) is the embedding loss that trains the semantic descriptor; \(L_{type}(P)\) is the type loss that trains the type vector; \(L_{param}(P)\) is the parameter loss that trains the primitive parameters. We set \(\alpha = 1.0\) and \(\beta = 0.1\) in our experiments. Network training employs ADAM [10]. In the following, we define each loss term.

#### Embedding loss

Similar to [1, 34], the embedding loss seeks to pull the semantic descriptors close to each other in the same primitive patch and push semantic descriptors of different primitive patches far from each other. Specifically, the loss consists of two terms: \(L_{pull}\) and \(L_{push}\). Denote \(P^k_{gt}\) as the ground-truth primitives. \(L_{pull}\) pulls each descriptor to the mean of the descriptors of the underlying primitive:

\begin{equation}
L_{pull} = \frac{1}{K} \sum_{k=1}^K \frac{1}{|P^k_{gt}|} \sum_{p_i \in P^k_{gt}} \max \left(\|d_i - d_{p_i}\| - \delta_i, 0\right), \tag{6}
\end{equation}

where \(d_{p_i} = \sum_{p_j \in P^k_{gt}} d_i / |P^k_{gt}|\). \(L_{push}\) pushes the embedding centers away from each other:

\begin{equation}
L_{push} = \frac{1}{K(K - 1)} \sum_{k < k'} \max \left(\delta_k - \|d_k - d_{k'}\|, 0\right). \tag{7}
\end{equation}

Combining (6) and (7), we define the embedding loss as

\begin{equation}
L_{emb} = \lambda L_{pull} + \nu L_{push}.
\end{equation}

In our experiments, \(\lambda = 1, \nu = 1, \delta_v = 0.5, \) and \(\delta_d = 1.5.\)
Type loss. We employ the cross-entropy loss $H_{ce}$ to define the type loss

$$\mathcal{L}_{type} = \frac{1}{n} \sum_{i=1}^{n} H_{ce}(t_i, t_i^g),$$

where $t_i^g$ is the ground-truth of $t_i$.

Parameter loss. The parameter loss employs the standard L2 loss on $\mathbb{R}^{22}$ between the predicted shape parameter $s_i$ and the underlying ground-truth $s_i^g$.

$$\mathcal{L}_{param} = \frac{1}{n} \sum_{i=1}^{n} \|s_i - s_i^g\|^2.$$

4.4.2 Learning of the HyperParameters

The second stage optimizes the hyperparameters of HPNet, including those used in defining the spectral modules and those used in defining the entropy terms (Equation 3). Since the total number of parameters is small, we use the established finite-difference approach for hyperparameter optimization from Song et al. [26]. This is done by optimizing the embedding loss on a validation dataset. Given the current hyperparameters, we compute the numerical gradient by sampling neighboring hyperparameter configurations. The best-fitting linear function gives the numerical gradient. We then apply a backtracking line search to determine the step-size. This gradient descent procedure terminates when the step size is smaller than $10^{-3}$, which typically occurs within 10-30 iterations.

5. Experimental Results

5.1. Experimental Setup

Datasets. We show experimental evaluation on two popular primitive segmentation datasets, i.e., ANSI Mechanical Component Dataset [14] and ABCParts Dataset [25]. ANSI mainly contains diverse mechanical components provided by TraceParts. Most of the objects in this dataset are composed by four basic primitives (plane, sphere, cylinder, and cone). We have 13k/3k/3k models on train/test/validation sets respectively. Each model contains 8192 points. ABCParts is derived from the ABC dataset [11], which provides a large source of 3D CAD models. In ABCParts, the objects are more complicated than those in ANSI, and each of them contains at least one B-spline surface patch. We have 24k/4k/4k models on train/test/validation sets on ABCParts and each model contains 10000 points. Please refer to supplementary materials for more details.

Evaluation metrics. We follow the standard metrics proposed by Li et al. [14] for quantitative evaluation.

- **Seg-IoU**: Denoting $K$ as the number of ground-truth patches, this metric evaluates the segmentation mean IoU score: 
  $$\frac{1}{K} \sum_{k=1}^{K} \text{IoU}(\mathbf{W}_{i,k}, \hat{\mathbf{W}}_{i,k}),$$
  where $\mathbf{W} \in \{0, 1\}^{n \times K}$ is the predicted segmentation membership matrix; $\hat{\mathbf{W}} \in \{0, 1\}^{n \times K}$ is the ground truth.

- **Type-IoU**: 
  $$\frac{1}{K} \sum_{k=1}^{K} \mathcal{I}[t_k = \hat{t}_k],$$
  where $t_k$ is the predicted primitive type for the $k$-th segment and $\hat{t}_k$ is the ground truth. $\mathcal{I}$ is an indicator function.
Table 2: Evaluation on different combinations of our components of HPNet.

<table>
<thead>
<tr>
<th></th>
<th>ANSI</th>
<th>ABCParts</th>
<th></th>
<th>ANSI</th>
<th>ABCParts</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Seg-IoU</td>
<td>Type-IoU</td>
<td></td>
<td>Seg-IoU</td>
<td>Type-IoU</td>
</tr>
<tr>
<td>Ours-nd</td>
<td>80.10</td>
<td>92.45</td>
<td></td>
<td>70.33</td>
<td>78.87</td>
</tr>
<tr>
<td>Ours-nc</td>
<td>92.92</td>
<td>98.53</td>
<td></td>
<td>83.80</td>
<td>89.41</td>
</tr>
<tr>
<td>Ours-ns</td>
<td>93.52</td>
<td>98.73</td>
<td></td>
<td>84.10</td>
<td>90.01</td>
</tr>
<tr>
<td>Ours-nw</td>
<td>94.07</td>
<td>98.90</td>
<td></td>
<td>84.78</td>
<td>90.56</td>
</tr>
<tr>
<td>Ours</td>
<td><strong>94.15</strong></td>
<td><strong>98.90</strong></td>
<td></td>
<td><strong>85.24</strong></td>
<td><strong>91.04</strong></td>
</tr>
</tbody>
</table>

Table 2: Evaluation on different combinations of our components of HPNet.

- **Res-Error**: \( \sum_{k=1}^{K} \frac{1}{m_k} \sum_{s_k \in \mathcal{P}_k} d_{\mathcal{P}_k}(\hat{s}_k) \), where \( \mathcal{P}_k \) is the predicted primitive path, \( M_k \) is the number of sampled points \( \hat{s}_k \) from ground truth primitive patch \( \mathcal{P}_k \), \( d_{\mathcal{P}_k}(\hat{s}_k) \) is the distance between \( \hat{s}_k \) and \( \mathcal{P}_k \).

- **P-coverage**: \( \frac{1}{n} \sum_{i=1}^{n} \delta \left( \min_{k=1}^{K} d_{\mathcal{P}_k}(p_i) < \epsilon \right) \), where \( \epsilon = 0.01 \).

5.2. Analysis of Results

Table 1 and Figure 3 present quantitative and qualitative results of HPNet. We can see that HPNet produces results that are close to the underlying ground-truth. Thanks to the smooth-edge module, the segment boundaries are smooth. Moreover, HPNet can even rectify small over-segmented primitive patches in the training data. Quantitatively, HPNet offers state-of-the-art results under all error metrics on these two datasets.

**Baseline comparison.** Our experimental study considers four baseline approaches. These include two state-of-the-art non-deep learning methods: nearest neighbor (NN) \([22]\) and Efficient RANSAC \([22]\), and two state-of-the-art deep learning methods: Supervised Primitive Fitting \(\text{SPFN}) [14]\) and ParseNet \([25]\). Note that to build a fair comparison, we replace the network backbones in SPFN and ParseNet to keep the same as HPNet.

Qualitatively (see Table 1), HPNet leads to considerable performance gains from all baseline approaches. Specifically, on ANSI, HPNet leads to salient 12.89% and 6.30% improvements in Seg-IOU under the point and point+normal settings, respectively. The improvements under other metrics are also considerable. For example, the Res-Error numbers decrease from 0.013 to 0.011 and from 0.010 to 0.008 under the point and point+normal settings, respectively. The improvements on ABCParts are also considerable, the improvements on Seg-IOU are 4.15% and 3.77% under the point and point+normal settings, respectively. HPNet also exhibits considerable improvements under other metrics.

The relative improvements on ANSI are bigger than those on ABCParts. As illustrated in Figure 4(a), we can understand this behavior from the fact that predictions of shape parameters of B-spline patches are less accurate than the other four primitive types.

5.3. Ablation Study

We proceed to study the impacts of different components of HPNet. Table 2 provides the overall Seg-IoU and Type-IoU scores after dropping each component of HPNet. Figure 4(b) shows Seg-IoU scores for different types of primitives.

**No descriptor module.** Table 2 shows that the descriptor module is critical for HPNet. Without this module, the Seg-IoU scores drop by 14.9% and 17.5% on ANSI and ABCParts, respectively. The Type-IoU scores drop by 6.5% and 13.4% on ANSI and ABCParts, respectively. Figure 4(b) shows that except for plane and sphere, the relative performance drops are glaring across other primitive types. This is expected as predicting accurate shape parameters requires global knowledge of the underlying primitive, which can be more difficult than predicting a semantic descriptor. The more significant drop on ABCParts
No smoothness module. As illustrated in Table 2, dropping the smoothness module leads to 0.67% and 1.34% decrements in the Seg-IoU scores and 0.17% and 1.13% decrements in the Type-IoU scores on ANSI and ABCParts, respectively. Figure 4(b) shows that the smoothness module contributes most to primitive types of cone, cylinder, and b-spline patches, which are likely to have sharp edges with other primitives. This also explains the smoothness module is slightly more effective on ABCParts than ANSI.

Figure 5 illustrates the effects of the smoothness module. Note that the smoothness module promotes smooth primitive boundaries. It also utilizes the sharp edges to merge and split primitives that are not possible when dropping this module.

No consistency module. As illustrated in Table 2, dropping the consistency module leads to 1.31% and 1.69% decrements in the Seg-IoU scores and 0.37% and 2.18% decrements in the Type-IoU scores on ANSI and ABCParts, respectively. Figure 4(b) shows that the consistency module contributes most to primitive types of Sphere, Cone, and Cylinder patches. One reason is that predictions of shape parameters tend to more accurate on these patches. On the other hand, the accuracy for Plane is already very high, leaving a small margin for improvements.

No weight learning. Finally, we study the importance of learning combination weights. Table 2 shows that using fixed combination weights leads to 0.08% and 0.46% decrements in the Seg-IoU scores and 0.00% and 0.48% decrements in the Type-IoU scores on ANSI and ABCParts, respectively. These statistics show that using adaptive weights is effective. Moreover, this strategy has larger impact on ABCParts than ANSI because the diversity of the primitives from ABCParts is larger than that from ANSI.

As shown in Figure 6, the learned weights are adaptive for different models. When the model contains small and complex primitives (shape(a)), the semantic descriptor is more useful. If the shape parameter and type predictions are accurate (shape(b)), the consistency descriptor places a more important role. Finally, when sharp edges are prominent (shape(c)), the smoothness descriptor becomes critical.

6. Conclusions and Limitations

In this paper, we have presented HPNet which combines multiple segmentation cues for primitive shape segmentation. Experimental results show that HPNet leads to considerable performance gains compared to previous approaches that leverage a single segmentation cue. Moreover, making the combination weights adaptive to the input models leads to additional performance improvement. Our ablation study further justifies different components of HPNet.

One limitation of HPNet is that it does not utilize symmetric relations among geometric primitives (c.f [15]). In the future, we plan to study novel graph neural networks to detect and enforce structural relations among primitives. Detecting such relations is essential for many models such as architectures.

Acknowledgements. Qixing Huang would like to acknowledge the support from NSF Career IIS-2047677 and NSF HDR TRIPODS-1934932. Etienne Vouga would like to acknowledge the support from NSF IIS-1910274, Side Effects Software Inc., and Adobe Inc.
References


[9] Zhizhong Kang and Zhen Li. Primitive fitting based on the efficient multibayesc algorithm. PLOS ONE, 10:1–21, 03 2015. 2


[26] Chen Song, Jiaru Song, and Qixing Huang. Hybridpose: 6d object pose estimation under hybrid representations. In


