

HPNet: Deep Primitive Segmentation Using Hybrid Representations

Siming Yan¹ Zhenpei Yang¹ Chongyang Ma² Haibin Huang²
 Etienne Vouga¹ Qixing Huang¹

¹The University of Texas at Austin ²Kuaishou 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

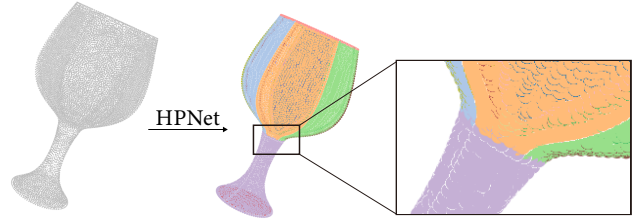


Figure 1: HPNet takes a point cloud as input and outputs detected primitive patches. It can handle diverse primitives at different scales. The detected primitives have smooth boundaries.

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 $\mathcal{P} = \mathcal{P}_1 \cup \dots \cup \mathcal{P}_K$ 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,

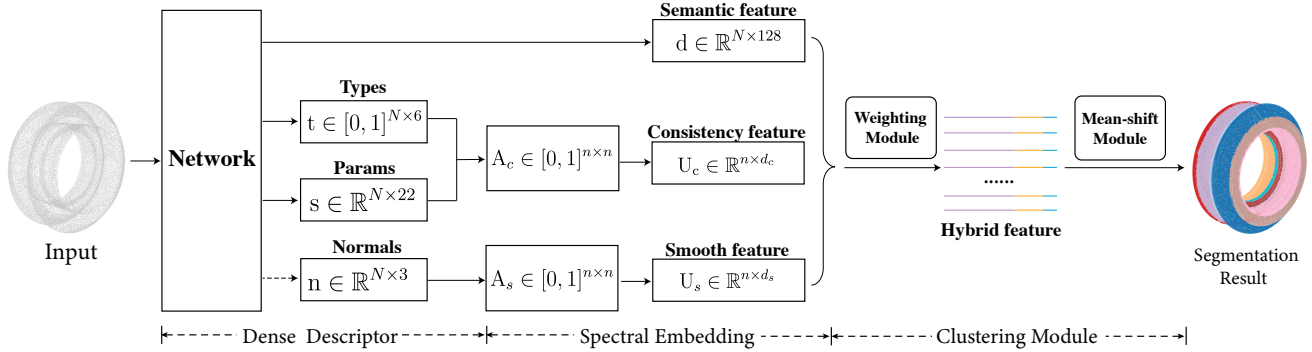


Figure 2: Overview of our approach pipeline. HPNet consists of three modules: (1) Dense Descriptor takes a point cloud and optional normal as input and outputs a semantic feature descriptor, a type indicator vector, and a shape parameter vector. (2) Spectral Embedding Module takes dense descriptors as input and builds geometric consistency matrix A_c and smoothness matrix A_s . Then it outputs consistency feature U_c and smoothness feature U_s . (3) Clustering Module combines three features with adaptive weights and use mean-shift clustering to output the segmentation result.

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 portion 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 $d(p_i, \mathbf{s}_j)$ between one point p_i and the predicted shape parameter \mathbf{s}_j associated with another point p_j [14]. Let t_j be the predicted primitive type of \mathbf{s}_j . When $t_j \in \{\text{Plane, Sphere, Cone, Cylinder}\}$, we define $d(p_i, \mathbf{s}_j) =$

$$\begin{cases} |\mathbf{p}_i^T \mathbf{n}_j - d_j| & t_j = \text{Plane} \\ \left| \|\mathbf{p}_i - \mathbf{o}_j\| - r_j \right| & t_j = \text{Sphere} \\ \left| \|(I - \mathbf{a}_j \mathbf{a}_j^T)(\mathbf{p}_i - \mathbf{o}_j)\| - r_j \right| & t_j = \text{Cylinder} \\ \|\mathbf{p}_i - \mathbf{o}_j\| \cos\left(\arccos\left(\mathbf{a}_j^T \frac{\mathbf{p}_i - \mathbf{o}_j}{\|\mathbf{p}_i - \mathbf{o}_j\|}\right) - \theta_j\right) & t_j = \text{Cone} \end{cases}$$

where \mathbf{n}_j and d_j denote the normal and distance of a plane primitive; \mathbf{o}_j and r_j the center and radius of a sphere primitive; \mathbf{a}_j , \mathbf{o}_j , and r_j the direction, center, and radius of a cylinder; and \mathbf{o}_j , \mathbf{a}_j , and θ_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 $d(p_i, \mathbf{s}_j)$ as the closest distance between p_i and the B-spline patch specified by \mathbf{s}_j . Please refer to the supplemental material for an efficient approach for computing $d(p_i, \mathbf{s}_j)$ approximately.

We then define the signed weight between p_i and \mathbf{s}_j as

$$w(p_i, \mathbf{s}_j) := \exp\left(-\frac{d^2(p_i, \mathbf{s}_j)}{2\sigma_{t_j}^2}\right),$$

where $\sigma_{t_j} > 0$ is a hyperparameter associated with type t_j . Finally, we define the consistency adjacency matrix $A_c \in [0, 1]^{n \times n}$, whose elements are given by

$$A_c(i, j) = (w(p_i, \mathbf{s}_j) + w(p_j, \mathbf{s}_i))/2, \quad 1 \leq i, j \leq n.$$

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 $U_c = (\sqrt{\frac{\lambda_1}{\lambda_1}} \mathbf{u}_{c,1}, \dots, \sqrt{\frac{\lambda_{d_c}}{\lambda_{d_c}}} \mathbf{u}_{c,d_c}) \in \mathbb{R}^{n \times d_c}$. Depending on the number of primitives, d_c varies across different shapes in our experiments.

Discussion. Below we provide an analysis to show that the spectral descriptor is superior to the predicted primitive parameters. Denote $A_c^{\mathbb{E}}$ as the matrix A_c with all entries (i, j) set to zero when p_i, p_j belong to different primitives. Without losing generality, we can reorder the vertices so that $A_c^{\mathbb{E}} = \text{diag}(A_{c,1}^{\mathbb{E}}, \dots, A_{c,K}^{\mathbb{E}})$ is a block diagonal matrix, where $A_{c,k}^{\mathbb{E}}$ contains the weights for pairs of points belonging to the k -th primitive. Decompose

$$A_c = A_c^{\mathbb{E}} + E.$$

To simplify the discussion, we further assume the weights are binary, i.e., 1 for consistent pairs and 0 for inconsistent pairs. We also assume $d_c = K$ in this analysis for convenience.

Let $U_c^{\mathbb{E}} \in \mathbb{R}^{n \times K}$ be the counterpart of U_c , whose columns are the re-scaled eigenvectors of $A_c^{\mathbb{E}}$. A variant of the Davis-Kahan theorem [33] provides the difference between $U_c^{\mathbb{E}}$ and U_c :

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

where $\|\cdot\|_{\mathcal{F}}$ denotes the Frobenius norm.

Applying (1), we argue that when primitive sizes are comparable, i.e., the size of the k -th primitive $n_k = \frac{n}{K}$,

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

where ρ is the fraction of outliers of the predicted shape parameters. Moreover, it is easy to see that corresponding error of the predicted shape parameters scales as $O(\sqrt{\rho})$. It follows when n is sufficiently large, the spectral descriptor is superior to the predicted shape parameters. Due to space constraint, we leave the proof to the supplemental material.

4.2.2 Smoothness Matrix

Consider a k -nearest neighbor graph $\mathcal{G} = (\mathcal{P}, \mathcal{E})$ whose vertices are taken from the input point cloud ($k = 50$ in this paper). For each edge $(i, j) \in \mathcal{E}$, we define the corresponding weight as

$$w_{(p_i, p_j)} = \exp\left(-\frac{\|\mathbf{n}_i - \mathbf{n}_j\|^2}{2\sigma_e^2}\right),$$

where σ_e is a hyperparameter, and where \mathbf{n}_i and \mathbf{n}_j are the normals at p_i and p_j , respectively. Let A_s be the weighted adjacency matrix of \mathcal{G} . The spectral descriptors U_s associated with A_s are then defined identically to U_c (columns are scaled leading eigenvectors of A_s).

Discussion. The usefulness of U_s comes from the fact the Euclidean distance between p_i and p_j defined by the rows of U_s is identical to the diffusion distance on the weighted graph \mathcal{G}_s specified by A_s (c.f. [27]). Note that points that are close to each other on the input model, but are on different sides of sharp edges, tend to have large diffusion distances (the paths connecting them have to detour around sharp edges). Therefore, these points have large distances in the embedding space.

4.3. Clustering Module

The clustering module concatenates the different types of point descriptors defined above, namely, the semantic descriptors \mathbf{d}_i and the two spectral features specified by $U_c^T \mathbf{e}_i$ and $U_s^T \mathbf{e}_i$. 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.

	Input	ANSI				ABCParts			
		seg iou	type iou	res error	P coverage	seg iou	type iou	res error	P coverage
NN	p	81.92	95.00	0.014	91.90	54.10	61.10	-	-
RANSAC [22]	p+n	70.10	93.13	0.029	78.79	67.21	-	0.022	83.40
SPFN [14]	p	77.61	95.43	0.014	92.10	58.15	73.88	0.023	87.58
SPFN [14]	p+n	88.05	98.10	0.011	92.94	73.41	80.04	0.020	89.40
ParseNet [25]	p	80.91	97.49	0.013	90.91	75.01	81.16	0.014	87.95
ParseNet [25]	p+n	88.57	98.26	0.010	92.72	82.14	88.60	0.011	92.97
Ours-wc	p	90.12	98.21	0.012	92.00	76.71	82.14	0.013	88.21
Ours-wc	p+n	92.41	98.87	0.010	93.58	83.20	89.54	0.010	93.15
Ours	p	91.34	98.66	0.011	93.02	78.12	85.32	0.012	90.54
Ours	p+n	94.15	98.90	0.008	94.02	85.24	91.04	0.009	94.31

Table 1: Benchmark evaluation on our approach and baseline approaches. We provide different input to the model: points(p) and points+normals(p+n). Here, Ours-wc stands for our method without combining two spectral descriptors.

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 $w_l \in (0, 1)$ for each feature, with $\sum_l w_l^2 = 1$. In the context of this paper, $L = 1 + d_c + d_s$, i.e., F_1 corresponds the semantic descriptors d_i , 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

$$P_{F_l}^{\sigma_l}(\mathbf{x}) := \frac{1}{n} (2\pi)^{-\frac{m_l}{2}} \sigma_l^{-m_l} \sum_{i=1}^n \exp\left(-\frac{\|\mathbf{x} - F_l^T \mathbf{e}_i\|^2}{2\sigma_l^2}\right),$$

where σ_l is a hyperparameter associated with F_l . The entropy of F_l is then

$$H_{\sigma_l}(F_l) := -\sum_{i=1}^n P_{F_l}^{\sigma_l}(F_l^T \mathbf{e}_i) \log\left(P_{F_l}^{\sigma_l}(F_l^T \mathbf{e}_i)\right). \quad (3)$$

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 $H_{\sigma_l}(F_l)$:

$$w_l := \bar{w}_l / \sqrt{\sum_l \bar{w}_l^2}, \quad \bar{w}_l := \frac{1}{H_{\sigma_l}(F_l)}. \quad (4)$$

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 \mathcal{P} . The total loss consists of three terms:

$$\mathcal{L}(\mathcal{P}) = \mathcal{L}_{\text{emb}}(\mathcal{P}) + \alpha \mathcal{L}_{\text{type}}(\mathcal{P}) + \beta \mathcal{L}_{\text{param}}(\mathcal{P}), \quad (5)$$

where $\mathcal{L}_{\text{emb}}(\mathcal{P})$ is the embedding loss that trains the semantic descriptor; $\mathcal{L}_{\text{type}}(\mathcal{P})$ is the type loss that trains the type vector; $\mathcal{L}_{\text{param}}(\mathcal{P})$ 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: $\mathcal{L}_{\text{pull}}$ and $\mathcal{L}_{\text{push}}$. Denote $\mathcal{P}_k^{\text{gt}}$ as the ground-truth primitives. $\mathcal{L}_{\text{pull}}$ pulls each descriptor to the mean of the descriptors of the underlying primitive:

$$\mathcal{L}_{\text{pull}} = \frac{1}{K} \sum_{k=1}^K \frac{1}{|\mathcal{P}_k^{\text{gt}}|} \sum_{p_i \in \mathcal{P}_k^{\text{gt}}} \max\left(\|d_i - d_{\mathcal{P}_k^{\text{gt}}}\| - \delta_1, 0\right), \quad (6)$$

where $d_{\mathcal{P}_k^{\text{gt}}} = \sum_{p_i \in \mathcal{P}_k^{\text{gt}}} d_i / |\mathcal{P}_k^{\text{gt}}|$. $\mathcal{L}_{\text{push}}$ pushes the embedding centers away from each other:

$$\mathcal{L}_{\text{push}} = \frac{1}{K(K-1)} \sum_{k < k'} \max\left(\delta_2 - \|d_{\mathcal{P}_k^{\text{gt}}} - d_{\mathcal{P}_{k'}^{\text{gt}}}\|, 0\right). \quad (7)$$

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

$$\mathcal{L}_{\text{emb}} = \lambda \mathcal{L}_{\text{pull}} + \nu \mathcal{L}_{\text{push}}.$$

In our experiments, $\lambda = 1$, $\nu = 1$, $\delta_v = 0.5$, and $\delta_d = 1.5$.

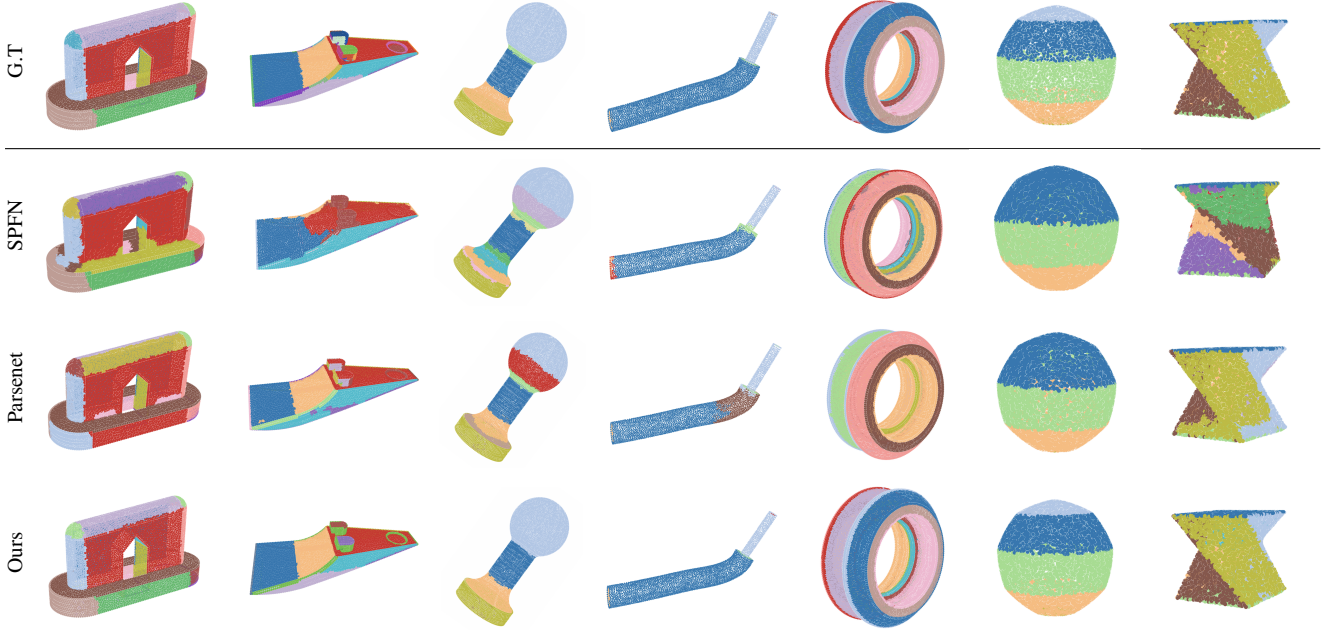


Figure 3: Primitive segmentation results with different methods. From top to down, we show the results of ground truth, SPFN [14], ParseNet [25], and Our approach.

Type loss. We employ the cross-entropy loss H_{ce} to define the type loss

$$\mathcal{L}_{\text{type}} = \frac{1}{n} \sum_{i=1}^n H_{ce}(\mathbf{t}_i, \mathbf{t}_i^{\text{gt}}),$$

where \mathbf{t}_i^{gt} is the ground-truth of \mathbf{t}_i .

Parameter loss. The parameter loss employs the standard L2 loss on \mathbb{R}^{22} between the predicted shape parameter \mathbf{s}_i and the underlying ground-truth \mathbf{s}_i^{gt} :

$$\mathcal{L}_{\text{param}} = \frac{1}{n} \sum_{i=1}^n \|\mathbf{s}_i - \mathbf{s}_i^{\text{gt}}\|^2.$$

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: $\frac{1}{K} \sum_{k=1}^K \text{IoU}(\mathbf{W}_{:,k}, \hat{\mathbf{W}}_{:,k})$, 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.

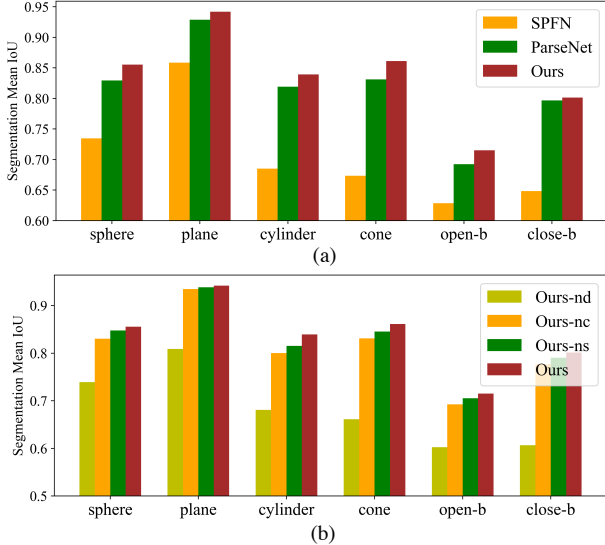


Figure 4: Mean IoU of segmentation results on different primitive types. Here, open-b and close-b represent open and closed B-spline patches. (a): comparison between HPNet and baseline methods. (b): comparison between different components of HPNet.

	ANSI		ABCParts	
	Seg-IoU	Type-IoU	Seg-IoU	Type-IoU
Ours-nd	80.10	92.45	70.33	78.87
Ours-nc	92.92	98.53	83.80	89.41
Ours-ns	93.52	98.73	84.10	90.01
Ours-nw	94.07	98.90	84.78	90.56
Ours	94.15	98.90	85.24	91.04

Table 2: Evaluation on different combinations of our approach. ‘Ours-nd’ denotes dropping the output of the descriptor module. ‘Ours-nc’ denotes dropping the output of the consistency module. ‘Ours-ns’ denotes dropping the output of the smoothness module. ‘Ours-nw’ means our approach without weight learning. ‘Ours’ corresponds to our full algorithm.

- *Res-Error*: $\sum_{k=1}^K \frac{1}{m_k} \sum_{\hat{s}_k \in \hat{\mathcal{P}}_k} d_{\mathcal{P}_k}(\hat{s}_k)$, 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 $\hat{\mathcal{P}}_k$, $d_{\mathcal{P}_k}(\hat{s}_k)$ is the distance between \hat{s}_k and \mathcal{P}_k .
- *P-coverage*: $\frac{1}{n} \sum_{i=1}^n \delta(\min_{k=1}^K d_{\mathcal{P}_k}(\mathbf{p}_i) < \epsilon)$, 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 sharp-edge module, the segment boundaries are smooth. Moreover, HPNet can even rectify small over-segmented

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) [23] and Efficient RANSAC [22], and two state-of-the-art deep learning methods: Supervised Primitive Fitting (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.

Quantitatively (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 consider improvements under other metrics.

Qualitatively (see Figure 3), HPNet leads to better results in the sense that it provides accurate segmentation for both large and small primitive patches. The segmentation boundaries are also smooth. In contrast, baseline approaches ParseNet and SPEN possesses non-smooth boundaries and the issues of over-segmentation and under-segmentation are noticeable.

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

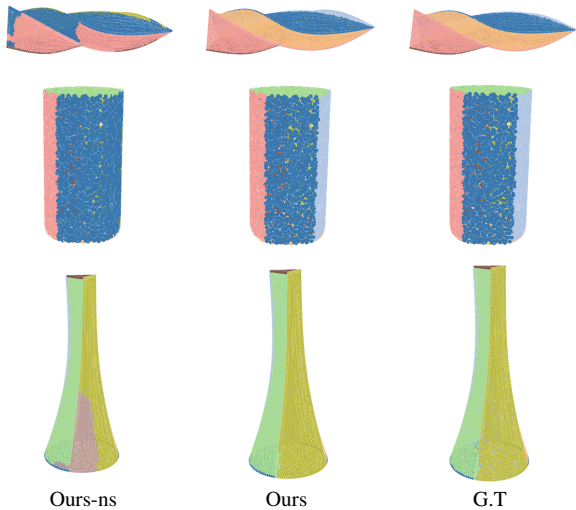


Figure 5: Examples on comparison between with and without sharp edge descriptor. Here, ‘Ours-ns’ represents our model without combining sharp edge descriptor. We notice that adding sharp edge descriptor helps model to capture boundary better.

than ANSI can be explained as the fact that ABCParts contain fewer primitives of types sphere and plane.

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

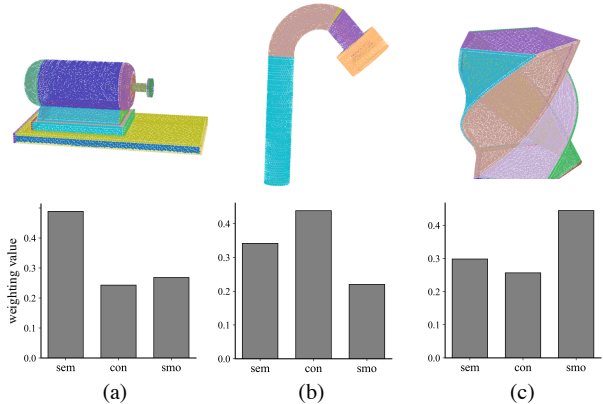


Figure 6: Illustration on the learned weights for three typical shapes. For simplicity, we use the L2 norm of the weighting vector of each type of spectral descriptors to represent each spectral embedding space. ‘sem’ denotes semantic descriptor. ‘con’ denotes geometric consistency descriptor. ‘smo’ denotes smoothness descriptor.

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 TRIPDS-1934932. Etienne Vouga would like to acknowledge the support from NSF IIS-1910274, Side Effects Software Inc., and Adobe Inc.

References

- [1] Bert De Brabandere, Davy Neven, and Luc Van Gool. Semantic instance segmentation with a discriminative loss function, 2017. [5](#)
- [2] Peter Carr, Yaser Sheikh, and Iain Matthews. Monocular object detection using 3d geometric primitives. In *Computer Vision – ECCV 2012*, 2012. [2](#)
- [3] Dorin Comaniciu and Peter Meer. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5):603–619, 2002. [5](#)
- [4] Manoranjan Dash and Huan Liu. Feature selection for clustering. In Takao Terano, Huan Liu, and Arbee L. P. Chen, editors, *Knowledge Discovery and Data Mining. Current Issues and New Applications*, pages 110–121, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg. [5](#)
- [5] Martin A Fischler and Robert C Bolles. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6):381–395, 1981. [2](#)
- [6] Leonidas J. Guibas, Qixing Huang, and Zhenxiao Liang. A condition number for joint optimization of cycle-consistent networks. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d’Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 1005–1015, 2019. [2](#)
- [7] Dirk Holz, Stefan Holzer, Radu Bogdan Rusu, and Sven Behnke. Real-time plane segmentation using rgb-d cameras. In *Robot Soccer World Cup*, pages 306–317. Springer Berlin Heidelberg, 2012. [2](#)
- [8] Adrien Kaiser, José Alonso Ybáñez Zepeda, and Tamy Boubekeur. A survey of simple geometric primitives detection methods for captured 3d data. *Comput. Graph. Forum*, 38(1):167–196, 2019. [2](#)
- [9] Zhizhong Kang and Zhen Li. Primitive fitting based on the efficient multibaysac algorithm. *PLOS ONE*, 10:1–21, 03 2015. [2](#)
- [10] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. [5](#)
- [11] Sebastian Koch, Albert Matveev, Zhongshi Jiang, Francis Williams, Alexey Artemov, Evgeny Burnaev, Marc Alexa, Denis Zorin, and Daniele Panozzo. Abc: A big cad model dataset for geometric deep learning, 2019. [6](#)
- [12] Florent Lafarge and Clément Mallet. Creating large-scale city models from 3D-point clouds: a robust approach with hybrid representation. *International Journal of Computer Vision*, 99(1):69–85, Feb. 2012. [2](#)
- [13] Jundong Li, Kewei Cheng, Suhang Wang, Fred Morstatter, Robert P. Trevino, Jiliang Tang, and Huan Liu. Feature selection: A data perspective. *ACM Comput. Surv.*, 50(6), Dec. 2017. [2](#)
- [14] 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. [1](#), [2](#), [4](#), [5](#), [6](#), [7](#)
- [15] Yangyan Li, Xiaokun Wu, Yiorgos Chrysathou, Andrei Sharf, Daniel Cohen-Or, and Niloy J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM Trans. Graph.*, 30(4), July 2011. [2](#), [8](#)
- [16] Huan Liu and Hiroshi Motoda. *Computational Methods of Feature Selection (Chapman & Hall/Crc Data Mining and Knowledge Discovery Series)*. Chapman & Hall/CRC, 2007. [2](#)
- [17] J Matas and O Chum. Randomized ransac with td,d test. *Image and Vision Computing*, 22(10):837–842, 2004. British Machine Vision Computing 2002. [2](#)
- [18] Sida Peng, Yuan Liu, Qixing Huang, Xiaowei Zhou, and Hujun Bao. Pvnnet: Pixel-wise voting network for 6dof pose estimation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 4561–4570. Computer Vision Foundation / IEEE, 2019. [3](#)
- [19] Omid Poursaeed, Matthew Fisher, Noam Aigerman, and Vladimir G. Kim. Coupling explicit and implicit surface representations for generative 3d modeling. 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 X*, volume 12355 of *Lecture Notes in Computer Science*, pages 667–683. Springer, 2020. [2](#)
- [20] Tahir Rabbani, Sander Dijkman, Frank van den Heuvel, and George Vosselman. An integrated approach for modelling and global registration of point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 61(6):355–370, 2007. [2](#)
- [21] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. *Computer Graphics Forum*, 26(2):214–226, June 2007. [2](#)
- [22] Ruwen Schnabel, Roland Wahl, and Reinhard Klein. Efficient ransac for point-cloud shape detection. *Computer graphics forum*, 26(2):214–226, 2007. [5](#), [7](#)
- [23] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhansu Maji. Csgnet: Neural shape parser for constructive solid geometry. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018. [1](#), [2](#), [7](#)
- [24] Gopal Sharma, Rishabh Goyal, Difan Liu, Evangelos Kalogerakis, and Subhansu Maji. CSGNET: Neural shape parser for constructive solid geometry. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5515–5523, 2018. [2](#)
- [25] Gopal Sharma, Difan Liu, Subhansu 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. [2](#), [3](#), [5](#), [6](#), [7](#)
- [26] Chen Song, Jiuru Song, and Qixing Huang. Hybridpose: 6d object pose estimation under hybrid representations. In

- 2020 *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2020, Seattle, WA, USA, June 13-19, 2020*, pages 428–437. IEEE, 2020. 6
- [27] Jian Sun, Maks Ovsjanikov, and Leonidas Guibas. A concise and provably informative multi-scale signature based on heat diffusion. In *Proceedings of the Symposium on Geometry Processing, SGP '09*, page 1383–1392, Goslar, DEU, 2009. Eurographics Association. 4
- [28] Jiliang Tang, Salem Alelyani, and Huan Liu. Feature selection for classification: A review. In Charu C. Aggarwal, editor, *Data Classification: Algorithms and Applications*, pages 37–64. CRC Press, 2014. 2
- [29] Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 2
- [30] 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. 3
- [31] Dong-Ming Yan, Wenping Wang, Yang Liu, and Zhouwang Yang. Variational mesh segmentation via quadric surface fitting. *Computer-Aided Design*, pages 1072–1082, April 2012. 2
- [32] Zhenpei Yang, Siming Yan, and Qixing Huang. Extreme relative pose network under hybrid representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2455–2464, 2020. 2
- [33] Yi Yu, Tengyao Wang, and Richard J. Samworth. A useful variant of the davis–kahan theorem for statisticians, 2014. 4
- [34] Zehao Yu, Jia Zheng, Dongze Lian, Zihan Zhou, and Shenghua Gao. Single-image piece-wise planar 3d reconstruction via associative embedding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 1029–1037, 2019. 5
- [35] Zaiwei Zhang, Zhenxiao Liang, Lemeng Wu, Xiaowei Zhou, and Qixing Huang. Path-invariant map networks. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*, pages 11084–11094. Computer Vision Foundation / IEEE, 2019. 2
- [36] Zaiwei Zhang, Bo Sun, Haitao Yang, and Qixing Huang. H3dnet: 3d object detection using hybrid geometric primitives. 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 XII*, volume 12357 of *Lecture Notes in Computer Science*, pages 311–329. Springer, 2020. 2
- [37] Zaiwei Zhang, Zhenpei Yang, Chongyang Ma, Linjie Luo, Alexander Huth, Etienne Vouga, and Qixing Huang. Deep generative modeling for scene synthesis via hybrid representations. *ACM Trans. Graph.*, 39(2):17:1–17:21, 2020. 2
- [38] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip Torr, and Vladlen Koltun. Point transformer, 2020. 3
- [39] Chuhan Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 900–909, 2017. 2