








PRIFIT: Learning to Fit Primitives Improves Few Shot Point Cloud Segmentation

G. Sharma^{†1} , B. Dash^{†1} , A. RoyChowdhury¹, M. Gadelha^{1,2} , M. Loizou³ , L. Cao¹, R. Wang¹,
E. G. Learned-Miller¹ , S. Maji¹  and E. Kalogerakis¹ 

¹University of Massachusetts Amherst ²Adobe ³University of Cyprus

Abstract

We present PRIFIT, a semi-supervised approach for label-efficient learning of 3D point cloud segmentation networks. PRIFIT combines geometric primitive fitting with point-based representation learning. Its key idea is to learn point representations whose clustering reveals shape regions that can be approximated well by basic geometric primitives, such as cuboids and ellipsoids. The learned point representations can then be re-used in existing network architectures for 3D point cloud segmentation, and improves their performance in the few-shot setting. According to our experiments on the widely used ShapeNet and PartNet benchmarks, PRIFIT outperforms several state-of-the-art methods in this setting, suggesting that decomposability into primitives is a useful prior for learning representations predictive of semantic parts. We present a number of ablative experiments varying the choice of geometric primitives and downstream tasks to demonstrate the effectiveness of the method.

CCS Concepts

• *Computing methodologies* → *Shape representations; Neural networks*; • *Theory of computation* → *Semi-supervised learning*;

1. Introduction

Several advances in visual recognition have become possible due to the supervised training of deep networks on massive collections of images. However, collecting manual supervision in 3D domains is more challenging, especially for tasks requiring detailed surface annotations e.g., part labels. To this end, we present PRIFIT, a *semi-supervised* approach for learning 3D point-based representations, guided by the decomposition of 3D shape into geometric primitives. Our approach exploits the fact that parts of 3D shapes are often aligned with simple geometric primitives, such as ellipsoids and cuboids. Even if these primitives capture 3D shapes at a rather coarse level, the induced partitions provide a strong prior for learning point representations useful for part segmentation networks, as seen in Fig. 1. This purely geometric task allows us to utilize vast amounts of unlabeled data in existing 3D shape repositories to guide representation learning for part segmentation. We show that the resulting representations are especially useful in the few-shot setting, where only a few labeled shapes are provided as supervision.

The overall framework for PRIFIT is based on a *point embedding* module and a *primitive fitting* module, as illustrated in Fig. 2. The point embedding module is a deep network that generates per-point embeddings for a 3D shape. Off-the-shelf networks can be used for

this purpose (e.g., PointNet++ [QYSG17], DGCNN [WSL*19]). The primitive fitting module follows a novel iterative clustering and primitive parameter estimation scheme based on the obtained per-point embeddings. It is fully differentiable, thus, the whole architecture can be trained end-to-end. The objective is to minimize a reconstruction loss, computed as the Chamfer distance between the 3D surface and the collection of fitted primitives. We experimented with various geometric primitives, including ellipsoids or cuboids due to their simplicity. We further considered parameterized geometric patches based on an Atlas [GFK*18], as an alternative surface primitive representation.

Our method achieves 63.4% part Intersection over Union (IoU) performance in ShapeNet segmentation dataset [CFG*15] with just one labeled example per-class, outperforming the prior state-of-the-art [GRS*20] by 1.6%. We also present results on the PartNet dataset [MZC*19a] where our approach provides 2.1% improvement compared to training from scratch approach while using 10 labeled examples per-class. We also present extensive analysis of the impact of various design choices, primitive types, size of the unlabeled dataset and different loss functions on the resulting shape segmentations. Our experiments indicate that the use of ellipsoids as geometric primitives provide the best performance, followed by cuboids, then AtlasNet patches, as is shown in Table 2.

[†] Equal contribution.

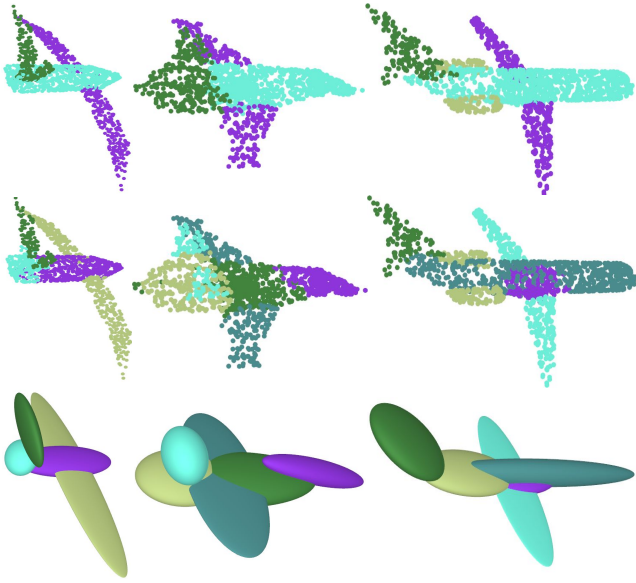


Figure 1: PRiFIT uses primitive fitting within a semi-supervised learning framework to learn 3D shape representations. Top row: 3D shapes represented as point clouds, where the color indicates the parts such as wings and engines. The induced partitions and shape reconstruction obtained by fitting ellipsoids to each shape using our approach are shown in the middle row and bottom row respectively. The induced partitions often have a significant overlap with semantic parts.

2. Related Work

We are interested in learning per-point representations of 3D shapes in a semi-supervised manner given a large number of unlabeled shapes and only a few labeled examples. To this end, we briefly review the literature on geometric primitive fitting and shape decomposition, few-shot learning, and deep primitive fitting. We also discuss the limitations of prior work and how we address them.

Geometric primitives and shape decomposition. Biederman’s recognition-by-components theory [Bie87] attempts to explain object recognition in humans by the ability to assemble basic shapes such as cylinders and cones, called *geons*, into the complex objects encountered in the visual world. Early work in cognitive science [HR83] shows that humans are likely to decompose a 3D shape along regions of maximum concavity, resulting in parts that tend to be convex, often referred to as the “*minima rule*”. Classical approaches in computer vision have modeled 3-D shapes as a composition of simpler primitives, *e.g.* work by Binford [BLM87; Bin71] and Marr [MN78]. More recent work in geometric processing has developed shape decomposition techniques that generate different types of primitives which are amenable to tasks like editing, grasping, tracking and animation [KYB19]. Those have explored primitives like 3D curves [GSMC09; MZL*09; GSV*17], cages [XLG12], sphere-meshes [TGB13], generalized cylinders [ZYH*15], radial basis functions [CBC*01; MGV11] and simple geometric primitives [SWK07]. This motivates the use

of our geometric primitive fitting as a self-supervised task for learning representations.

Unsupervised learning for 3D data. Several previous techniques have been proposed to learn 3D representations without relying on extra annotations. Many such techniques rely on reconstruction approaches [YFST18; GFK*18; GWM18; ZBDT19; YHH*19]. FoldingNet [YFST18] uses an auto-encoder trained with permutation invariant losses to reconstruct the point cloud. Their decoder consists of a neural network representing a surface parametrized on a 2D grid. AtlasNet [GFK*18] proposes using several such decoders that result in the reconstructed surface being represented as a collection of surface patches. Li *et al.* [LCL18] presents SO-Net that models spatial distribution of point cloud by constructing a self-organizing map, which is used to extract hierarchical features. The proposed architecture trained in auto-encoder fashion learns representation useful for classification and segmentation. Chen *et al.* [CYF*19] propose an auto-encoder (BAE-Net) with multiple branches, where each branch is used to reconstruct the shape by producing implicit fields instead of point clouds. However, this requires one decoder for separate part, which restricts its use to category-specific models. In contrast, our approach can train a single network in a category-agnostic manner because our approach is based on inducing convexity priors in the embedding through primitive fitting – this does not require category-specific knowledge, such as the number of semantic parts, making our approach more general.

Yang *et al.* [YC21] also propose fitting cuboids to point clouds for the task of co-segmentation. In order to segment the input point cloud based on the fitted cuboids, they define point to primitive membership based on point embeddings and enforce this membership to give correct correspondence between points and primitives. However, their method trains a category-specific model, thus suffers from the same limitation as BAE-Net.

Other techniques proposed models for generating implicit functions from point clouds [GCV*19; DGY*20; MON*19], but it is unclear how well the representations learned by those methods perform in recognition tasks. Several works use reconstruction losses along with other self supervised tasks. Hassani *et al.* [HH19] propose multiple tasks: reconstruction, clustering and classification to learn point representation for shapes. Thabet *et al.* [TAG19] propose a self-supervision task of predicting the next point in a space filling curve (Morton-order curve) using RNN. The output features from the RNN are used for semantic segmentation tasks. Several works have proposed learning point representation using noisy labels and semantic tags available from various shape repositories. Sharma *et al.* [SKM19] learn point representations using noisy part hierarchies and designer-labeled semantic tags for few-shot semantic segmentation. Muralikrishnan *et al.* [MKC18] design a U-Net to learn point representations that predicts user-prescribed shape-level tags by first predicting intermediate semantic segmentation. More recently, Xie *et al.* [XGG*20] learn per-point representation for 3D scenes, where point embeddings of matched points from two different views of a scene are pushed closer than un-matched points under a contrastive learning framework. Sun *et al.* [STD*21] propose an approach for learning shape representations by canoni-

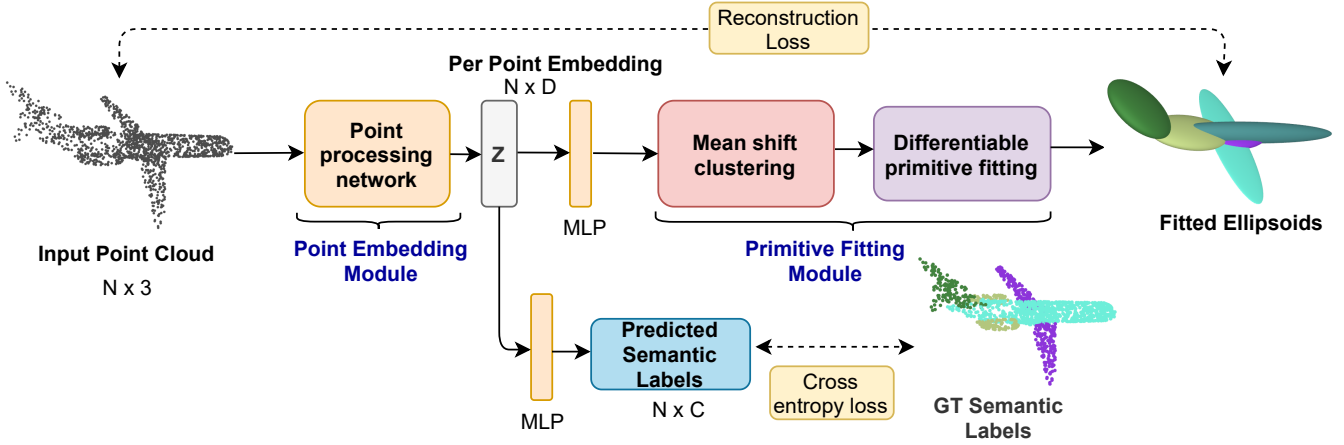


Figure 2: Overview of PRIFIT. Given a point cloud, the point-embedding module outputs a feature representation for each point. This is processed through the primitive-fitting module, that uses mean-shift clustering to cluster the points and fit a geometric primitive to each cluster. We train the network with a reconstruction loss between the fitted primitives and the input point cloud over the unlabeled shapes, and a categorical cross entropy loss over a small number of labeled shapes.

calizing point clouds with the help of capsule network while simultaneously decomposing point clouds into parts.

Closely related to our work, Gadelha et al. [GRS*20] use approximate convex decomposition of watertight meshes as source of self-supervision by training a metric over point clouds that respect the given decomposition. Our approach directly operates on point clouds and integrates the decomposition objectives in a unified and end-to-end trainable manner. Empirically we observe that this improves performance. It also removes the need for having a black-box decomposition approach that is separated from the network training.

Semi-supervised learning for 3D data. Similar to our approach, Alliegro et al. [ABT20] use joint supervised and self-supervised learning for learning 3D shape features. Their approach is based on solving 3D jigsaw puzzles as a self-supervised task to learn shape representation for classification and part segmentation. Wang et al. [WLF20] proposed semi-supervised approach that aligns a labeled template shape to unlabeled target shapes to transfer labels using learned deformation. Luo et al. [LMH*20] proposed discovering 3D parts for objects in unseen categories by extracting part-level features through encoding their local context, then using agglomerative clustering and a grouping policy to merge small part proposals into bigger ones in a bottom-up fashion. Our method follows an orthogonal approach where features are learned and clustered into parts guided by primitive fitting.

Deep primitive fitting. Several approaches have investigated the use of deep learning models for shape decomposition. Their common idea is to learn point-level representations used to generate primitives. Several primitive types have been proposed, including superquadrics [PvGG20; PUG19], cuboids [GGC*20; TSG*17], generalized cylinder [LGB*21] and radial basis functions [GCV*19]. However, all these approaches have focused on generative tasks with the goal of editing or manipulating a 3D

shape. Our insight is that reconstructing a shape by assembling simpler components improves representation learning for discriminative tasks, especially when only a few labeled training examples are available.

3. Method

Our method assumes that one is provided with a small set of labeled shapes \mathcal{X}_l and a large set of unlabeled shapes \mathcal{X}_u . Each shape $X \in \{\mathcal{X}_l, \mathcal{X}_u\}$ is represented as a point cloud with N points, i.e., $X = \{x_i\}$ where $x_i \in \mathbb{R}^3$. The shapes in \mathcal{X}_l additionally come with part label $Y = \{y_i\}$ for each point. In our experiments we use the entire set of shapes from the ShapeNet core dataset [CFG*15] and few labeled examples from the ShapeNet semantic segmentation dataset and PartNet dataset.

The architecture of PRIFIT consists of a *point embedding* module Φ and a *primitive fitting* module Ψ . The point embedding module $\Phi(X)$ maps the shape into embeddings corresponding to each point $\{\Phi(x_i)\} \in \mathbb{R}^D$. The primitive fitting module Ψ maps the set of point embeddings to a set of primitives $\{P_i\} \in \mathcal{P}$. Thus $\Psi \circ \Phi : X \rightarrow \mathcal{P}$ is a mapping from point clouds to primitives. In addition the point embeddings can be mapped to point labels via a classification function Θ and thus, $\Theta \circ \Phi : X \rightarrow Y$. We follow a *joint training* approach where shapes from \mathcal{X}_l are used to compute a supervised loss and the shapes from \mathcal{X}_u are used to compute a self-supervised loss for learning by minimizing the following objective:

$$\min_{\Phi, \Psi, \Theta} \mathcal{L}_{\text{ssl}} + \mathcal{L}_{\text{sl}}, \text{ where} \quad (1)$$

$$\mathcal{L}_{\text{ssl}} = \mathbb{E}_{X \sim \mathcal{X}_u} [\ell_{\text{ssl}}(X, \Psi \circ \Phi(X))], \text{ and} \quad (2)$$

$$\mathcal{L}_{\text{sl}} = \mathbb{E}_{(X, Y) \sim \mathcal{X}_l} [\ell_{\text{sl}}(Y, \Theta \circ \Phi(X))]. \quad (3)$$

Here ℓ_{ssl} is defined as a *self-supervision loss* between the point

cloud and a set of primitives, while ℓ_{sj} is the cross entropy loss between predicted and true labels. We describe the details of the point embedding module in Sec.3.1 and the primitive fitting module in Sec. 3.2. Finally in Sec. 3.3 we describe various loss functions used to train PRIFIT.

3.1. Point embedding module

This module produces an embedding of each point in a point cloud. While any point cloud architecture [QSMG17; QYSG17; WSL*19] can be used, we experiment with PointNet++ [QYSG17] and DGCNN [WSL*19], two popular architectures for point cloud segmentation. These architectures have also been used in prior work on few-shot semantic segmentation making a comparison easier.

3.2. Primitive fitting module

The primitive fitting module is divided into a *decomposition* step that groups the set of points into clusters in the embedding space, and a *fitting* step that estimates the parameters of the primitive for each cluster.

Decomposing a point cloud. These point embeddings $\Phi(x_i) \in \mathbb{R}^D$ are grouped into M clusters using a differentiable mean-shift clustering. The motivation behind the choice of mean-shift over other clustering approaches such as k-means is that it allows the number of clusters to vary according to a kernel bandwidth. In general we expect that different shapes require different number of clusters. We use recurrent mean-shift updates in a differentiable manner as proposed by [KF18]. Specifically, we initialize seed points as $G^{(0)} = Z \in \mathbb{R}^{N \times D}$ and update them as follows:

$$G^{(t)} = KZD^{-1}. \quad (4)$$

We use the von Mises-Fisher kernel [MJ09] $K = \exp(G^{(t-1)}Z^T/b^2)$, where $D = \text{diag}(K\mathbb{1})$ and b is the bandwidth. The bandwidth is computed dynamically for each shape by using the average distance of each point to its 100th neighbor in the embedding space [Läu88]. K is updated after every iteration. The embeddings are normalized to unit norm, i.e., $\|z_i\|_2 = 1$, after each iteration. We perform 10 iterations during training. After these updates, a non-max suppression step yields M cluster centers $c_{m,m} = \{1, \dots, M\}$ while making sure number of clusters are bounded. The non-max suppression is done as follows: we start by extracting the highest density points that include at least one other point within a prescribed radius b . Then we remove all points within that radius and repeat until no other high density points are left. All selected high density points in this way act as cluster centers.

Having updated the embeddings with the mean-shift iterations, we can now define a *soft membership* W for each point x_i , represented by the embedding vector g_i^\dagger , to the cluster center c_m as

$$w_{i,m} = \frac{\exp(c_m^\top g_i)}{\sum_m \exp(c_m^\top g_i)}, \quad (5)$$

[†] Superscript t is dropped.

where $w_{i,m} = 1$ represents the full membership of the i^{th} point to the m^{th} cluster.

Ellipsoid fitting. Given the clustering, we then fit an ellipsoid to each of the clusters. Traditionally, fitting an ellipsoid to a point cloud is formulated as a *minimum volume enclosing ellipsoid* [FP93] and solved using the Khachiyan algorithm. However, it involves an optimization procedure that is not readily differentiable, thus, making it hard to incorporate in an end-to-end training pipeline. It is also susceptible to outliers (see supplementary materials for details). We instead rely on a simpler and faster differential approximation based on singular value decomposition (SVD) for ellipsoid fitting.

Given the membership of the point to m^{th} cluster we first center the points and compute the SVD as

$$\mu = W_m X Z^{-1}, \quad (6)$$

$$\bar{X} = X - \mu, \text{ and} \quad (7)$$

$$U, S, V = \text{SVD}(\bar{X}^T W_m \bar{X} Z^{-1}). \quad (8)$$

Here W_m is the diagonal matrix with $w_{i,m}$ as its diagonal entries ($W_m[i,i] = w_{i,m}$ for m^{th} cluster) and $Z = \text{trace}(W_m)$. The orientation of an ellipsoid is given by V . The length of the principal axes can be computed from the singular values as $a_i = \kappa \sqrt{S_{ii}}$. We select $\kappa = \sqrt{3}/2$ by cross validation. The matrix W_m selects the points with membership to the m^{th} cluster in a ‘soft’ or weighted fashion, and the SVD in Eq. 8 gives us the parameters of the ellipsoid that fits these weighted points.

Discussion — alternate choices for primitives. Our approach can be used to fit cuboids instead of ellipsoids by considering the bounding box of the fitted ellipsoids instead. This may induce different partitions over the point clouds, and we empirically compare its performance in the Sec. 4. Another choice is to represent the surface using an Atlas – a collection of parameterized patches. We use the technique proposed in AtlasNet [GFK*18] where neural networks f_θ parameterize the coordinate charts $f_\theta : [0, 1]^2 \rightarrow (x, y, z)$ conditioned on a latent code computed from the point cloud. The decoders are trained along with the encoder using gradient descent to minimize Chamfer distance between input points and output points across all decoders. An encoder trained in this fashion learns to decompose input points into complex primitives, i.e. via arbitrary deformations of the 2D plane. Point representations learnt in this fashion by the encoder can be used for downstream few-shot semantic segmentation task as shown in Sec. 4 and Tab. 2. However, this approach requires adding multiple decoder neural networks. Our *ellipsoid fitting* approach does not require significant architecture changes and avoids the extra parameters required by AtlasNet.

3.3. Loss functions

Reconstruction loss. This is computed as the Chamfer distance of input point clouds from predicted primitives. For the distance of a point on the input surface to the predicted surface we use

$$\mathcal{L}_1 = \sum_{i=1}^N \min_m D_m^2(x_i). \quad (9)$$

Here N is the number of points on the input shape and $D_m(x_i)$ is the distance of input point x_i from the m^{th} primitive. This is one side of Chamfer distance, ensuring that the predicted primitives cover the input surface. In order to compute the distance of a point x from an ellipsoid or cuboid primitive, first the point is centered and re-oriented using the center and principal axes computed for the primitive in Eq. 6 and Eq. 8 respectively: $p = V^T(x - \mu)$. Then we compute the signed distance $SD(p)$ of the point to each primitive, as described in the next paragraphs. Note that the unsigned distance used in Eq. 9 is simply computed as $D(p) = |SD(p)|$.

In the case of an ellipsoid primitive, we compute the approximate signed distance [Qui] of the transformed point $p = (p_x, p_y, p_z)$ to it as

$$SD(p) = k_1(k_1 - 1)/k_2, \quad (10)$$

where k_1 and k_2 are calculated as

$$k_1 = \sqrt{\left(\frac{p_x}{s_x}\right)^2 + \left(\frac{p_y}{s_y}\right)^2 + \left(\frac{p_z}{s_z}\right)^2} \text{ and} \quad (11)$$

$$k_2 = \sqrt{\left(\frac{p_x}{s_x^2}\right)^2 + \left(\frac{p_y}{s_y^2}\right)^2 + \left(\frac{p_z}{s_z^2}\right)^2} \quad (12)$$

Here $s = (s_x, s_y, s_z)$ is a vector storing the lengths of the ellipsoid semi-axis in all 3 directions.

In the case of a cuboid primitive, the distance of the transformed point p to the primitive is computed as follows:

$$q = (|p_x| - s_x, |p_y| - s_y, |p_z| - s_z), \quad (13)$$

$$q_+ = (\max\{q_x, 0\}, \max\{q_y, 0\}, \max\{q_z, 0\}), \text{ and} \quad (14)$$

$$SD(p) = \|q_+\|_2 + \min\{\max\{q_x, q_y, q_z\}, 0\}. \quad (15)$$

Here $s = (s_x, s_y, s_z)$ stores here the half-axes lengths of the cuboid. Note that the above equations measuring distances of points to primitives are analytic and differentiable with respect to the point coordinates.

To ensure that the input surface covers the predicted primitives we minimize the loss

$$\mathcal{L}_2 = \sum_{m=1}^M \sum_{p \sim E_m} \min_{i=1}^N \|x_i - p\|_2^2, \quad (16)$$

where E_m is the m^{th} fitted primitive. We sample 10k points over all ellipsoids, weighted by the surface area of each primitive. We uniformly sample each primitive surface. These point samples are a function of the parameters of the predicted primitives and hence

the gradients of the loss function can be back-propagated to the network. In the case of an ellipsoid, its parametric equation is:

$$(x, y, z) = (s_x \cos u \sin v, s_y \sin u \sin v, s_z \cos v), \quad (17)$$

and its inverse parameterization is:

$$(v, u) = (\arccos(s_z/C), \text{atan2}(s_y/B, s_x/A)), \quad (18)$$

where (u, v) are the parameters of the ellipsoid's 2D parametric domain, (x, y, z) the point coordinates, and (s_x, s_y, s_z) the lengths of semi-axis of the ellipsoid. To sample the ellipsoid in a near-uniform manner, we start by creating a standard axis-aligned and origin centered mesh using the principal axis lengths predicted by PRIFIT. Then we apply Poisson surface sampling to gather points on the surface in an approximately uniform manner. We then compute parameters (u, v) of the sampled points using Eq. 18. Note that this is done outside the computation graph. We inject the computed parameters back to the computation graph using Eq. 17 to compute point coordinates (x, y, z) again. These point coordinates are rotated and shifted based on the predicted axis and center respectively. Sampling over the cuboid surfaces is done in a similar manner.

We use the two-sided loss to minimize reconstruction error

$$\ell_{recon} = \mathcal{L}_1 + \mathcal{L}_2. \quad (19)$$

The hypothesis is that for a small number of primitives the above losses encourage the predicted primitives to fit the input surface. Since the fitting is done using a union of convex primitives, each diagonal entry of the matrix W_m in Eq. 8 should have higher weights to sets of points that belong to convex regions, thereby resulting in a convex (or approximately convex) segmentation of a point cloud. The point representations learnt in this manner are helpful for point cloud segmentation as shown in Table 2.

Intersection loss. To encourage spatially compact clusters we introduce a loss function that penalizes overlap between ellipsoids. Note that the clustering objective does not guarantee this as they operate on an abstract embedding space. Specifically, for each point p sampled inside the surface of predicted shape should be contained inside a single primitive. Alternatively the corresponding primitive should have negative signed distance $S_m(p)$ at that point p , whereas the signed distance (SD) should be positive for the remaining primitives. Let \mathcal{V}_m be the set of points sampled inside the primitive m . Then intersection loss is defined as

$$\ell_{inter} = \sum_m \sum_{p \sim \mathcal{V}_m} \sum_{j \neq m} [S_j(p)]_-^2, \quad (20)$$

where $[S_m(p)]_- = \min(S_m(p), 0)$ includes only the points with negative SD, as points with positive SD are outside the primitive and do not contribute in intersection. We use a differentiable approximation of the signed distances to ellipsoid and cuboid as described in Eq. 10 and Eq. 15. In the Table 5 we show that including intersection loss improves semantic segmentation performance.

Similarity loss. Due to the general initialization schemes of the network weights, all the per-point embeddings are similar, which leads to mean-shift clustering grouping all points into a single cluster. This trivial clustering results in a single ellipsoid fitted to the

entire shape which is similar to the PCA of the point cloud. This local minima results in negligible gradients being back-propagated to the network and prevents learning of useful features. We observed this phenomenon in several point-based network architectures. This local minima can be avoided by spreading the point embeddings across the space. We add a small penalty only at the early stage of training to minimize the similarity of output point embeddings G from mean-shift iterations (Eq. 4) as

$$\ell_{sym} = \sum_{i \neq j} (1 + g_i g_j^T)^2. \quad (21)$$

We discuss the quantitative effect of this loss in the Section 4.

3.4. Training details

We train our network jointly using both a self-supervised loss and a supervised loss. We alternate between our self-supervised training while sampling point clouds from the entire unlabeled \mathcal{X}_u dataset and supervised training while taking limited samples from the labeled \mathcal{X}_l set. Hence our joint supervision and self-supervision approach follows semi-supervised learning paradigm.

$$L = \underbrace{\sum_{X \sim \mathcal{X}_u} \ell_{recon} + \lambda_1 \ell_{inter} + \lambda_2 \ell_{sym}}_{\ell_{ssl}(\text{self-supervision})} + \underbrace{\sum_{X \sim \mathcal{X}_l} \ell_{ce}}_{\ell_{sl}(\text{supervision})} \quad (22)$$

where ℓ_{ce} is a cross entropy loss, $\lambda_1 = 0.001$ and $\lambda_2 = 2$ are constants. To produce segmentation labels for each point we implement the classification function Θ in Eq. 3 using a 1D convolution layer followed by a softmax function. More implementation details are provided in the supplementary material.

Back-propagation and numerical stability

- To back-propagate the gradients through SVD computation we use analytic gradients derived by Ionescu *et al.* [MN99]. We implemented a custom Pytorch layer following [LSD*18]. SVD of a matrix $X \in \mathbb{R}^{m,n}$ is given by $X = U\Sigma V^T$ with $m \geq n$, $U^T U = I$, $V^T V = I$ and $\Sigma \in \mathbb{R}^{n,n}$ possessing diagonal structure. We define an SVD layer that receives a matrix X as input and produces a tuple of 3 matrices U , Σ and V , defined as $f(X) = (U, \Sigma, V)$. Given a loss function $\mathcal{L} = L(f(X))$, we are interested in $\frac{\partial \mathcal{L}}{\partial X}$. Assuming, $\frac{\partial \mathcal{L}}{\partial U}$, $\frac{\partial \mathcal{L}}{\partial \Sigma}$, $\frac{\partial \mathcal{L}}{\partial V}$ are known using automatic differentiation. For our purpose, we assume that $\frac{\partial \mathcal{L}}{\partial U} = 0$ as U is not a part of the computation graph. Then gradient of the output w.r.t. the input X is given by

$$\frac{\partial \mathcal{L}}{\partial X} = U \left(\frac{\partial \mathcal{L}}{\partial \Sigma} \right)_{diag} V^T + 2U\Sigma(K^T \circ (V^T \left(\frac{\partial \mathcal{L}}{\partial V} \right)))_{sym} V^T, \quad (23)$$

$$K_{ij} = \begin{cases} \frac{1}{\sigma_i^2 - \sigma_j^2}, & i \neq j \\ 0, & i = j. \end{cases} \quad (24)$$

σ_i is the i^{th} singular value. $M_{sym} = \frac{1}{2}(M + M^T)$ and M_{diag} is M with off diagonal entries set to 0. When back-propagating gradients through SVD, gradients can go to infinity when singular values are indistinct. This happens when membership weights

in a cluster are concentrated on a line, point or sphere. Following [IVS15], the above term K_{ij} is changed to $K_{ij} = 1/(\sigma_i + \sigma_j) \text{sign}(\sigma_i - \sigma_j) (\max(|\sigma_i - \sigma_j|, \epsilon))$ with $\epsilon = 10^{-6}$. SVD computation can still be unstable when the condition number of the input matrix is large. In this case, we remove that cluster from the backward pass when condition number is greater than 10^5 .

- *Membership function:* In Eq. 5 the input to the exponential function is clamped to avoid numerical instability during training. Furthermore, the quantity $r = \max_{i,m} c_m^T g_i$ is subtracted from the arguments of exponential function in both numerator and denominator to avoid weights becoming infinity.
- *Differentiability of mean shift clustering procedure:* The membership matrix $W \in \mathbb{R}^{N \times M}$ is constructed by doing non-max suppression (NMS) over output G of mean shift iteration. The derivative of NMS w.r.t embeddings G is either zero or undefined, thereby making it non-differentiable. Thus we remove NMS from the computation graph and back-propagate through the rest of the graph, which is differentiable through Eq. 5. This can be seen as a straight-through estimator [SHWA15], which has been used in previous shape parsing works [LSD*18; SLM*20].

The code of our implementation can be found on the web page: <https://hippogriff.github.io/prifit/>.

4. Experiments

4.1. Datasets

As a source of unlabeled data for the task of self supervision, we use the ShapeNet Core dataset [CFG*15], which consists of 55 categories with 55,447 meshes in total. We sample these meshes uniformly to get 2048 points per shape. For the task of few-shot semantic segmentation, we use the ShapeNet Semantic Segmentation dataset, which consists of 16,881 labeled point clouds across 16 shape categories with total 50 part categories.

We also evaluate our method on the PartNet dataset [MZC*19b]. This dataset provides *fine-grained* semantic segmentation annotation for various 3D shape categories, unlike the more coarse-level shape parts in the ShapeNet dataset. We use 12 categories from “level-3”, which denotes the finest level of segmentation. For training different approaches in few-shot framework, we remove test shapes of labeled dataset \mathcal{X}_l from our self-supervision dataset \mathcal{X}_u . This avoids train-test set overlap.

4.2. Few-shot part segmentation on ShapeNet

For each category from the ShapeNet part segmentation dataset, we randomly sample k labeled shapes ($16 \times k$ in total) and use these for training the semantic segmentation component of our model. We use the entire training set from the ShapeNet core dataset for self supervision task. We train a *single* model on all 16 shape categories of ShapeNet.

Baselines. Our first baseline takes the PointNet++ as the base architecture and trains it from scratch on only labeled training examples, using k labeled shapes per category in a few-shot setup. Second, we create a reconstruction baseline – we use a PointNet++

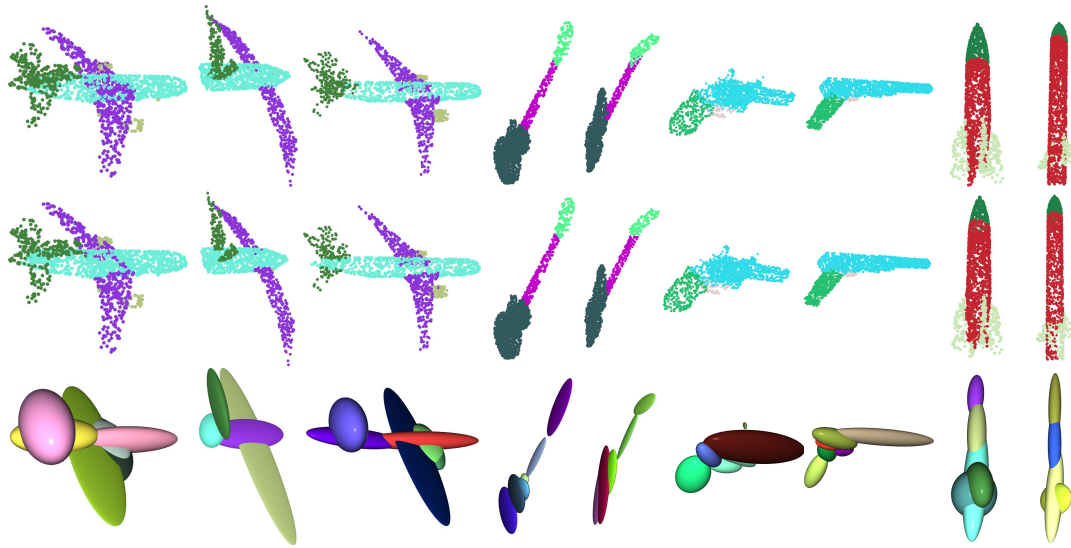


Figure 3: Visualization of predicted semantic labels and ellipsoids on the ShapeNet dataset. Top: Ground truth point clouds, **middle:** predicted labels using our fitting approach, trained using $k = 10$ labeled examples per category, **bottom:** predicted ellipsoids. PRiFIT predicts variable number of ellipsoids to approximate the input point cloud while maintaining correspondence with semantic parts.

Samples/cls.	k=1	k=3	k=5	k=10	k=20	k=50	k=100	k=max
Baseline	53.15 \pm 2.4	59.54 \pm 1.4	68.14 \pm 0.9	71.32 \pm 0.5	75.22 \pm 0.8	78.79 \pm 0.4	79.67 \pm 0.3	81.40 \pm 0.4
PRiFIT	63.14 \pm 3.4	71.24 \pm 1.3	73.75 \pm 0.7	75.03 \pm 0.9	76.73 \pm 0.5	79.28 \pm 0.2	80.16 \pm 0.2	80.40 \pm 0.1

Table 1: Few-shot segmentation on the ShapeNet dataset (class avg. IoU over 5 rounds). The number of shots or samples per class is denoted by k for each of the 16 ShapeNet shape categories used for supervised training. Our proposed method PRiFIT consistently outperforms the baselines.

as a shared feature extractor, which extracts a global shape encoding that is input to an AtlasNet decoder [GFK*18] with 25 charts. A separate decoder is used to predict per-point semantic labels. This network is trained using a Chamfer distance-based reconstruction loss using the entire unlabeled training set and using k labeled training examples. We use 5 different rounds with sampled labeled sets at various values of k and report their average performance. We train a single model for all categories.

Discussion of results. Table 1 shows results on few-shot segmentation at different number of labeled examples. Our method PRiFIT performs better than the supervised baseline showing the effectiveness of our method as semi-supervised approach.

In Table 2 we compare our approach with previous methods using instance IOU and class IOU [QYSG17], using 1% and 5% of the labeled training set to train different methods. Note that instance IOU is highly influenced by the shape categories with large number of testing shapes *e.g.* Chair, Table. Class IOU, on the other hand gives equal importance to all categories, hence it is a more robust evaluation metric. PRiFIT performs better than previous self-supervised [TAG19; GRS*20; HH19] and semi-supervised [ABT20; WLF20] approaches. Notable our approach significantly outperforms learned deformation based approach pro-

Method	1%	5%	1%	5%
	ins IoU	ins IoU	cls IoU	cls IoU
SO-Net [LCH18]	64.0	69.0	-	-
PointCapsNet [ZBDT19]	67.0	70.0	-	-
MortonNet [TAG19]	-	77.1	-	-
Deformation [WLF20]	68.9	-	66.2	-
JointSSL [ABT20]	71.9	77.4	-	-
Multi-task [HH19]	68.2	77.7	-	72.1
ACD [GRS*20] †	75.1	78.6	74.6	77.5
PRiFIT w/ Atlas	73.8	78.6	74.5	78.9
PRiFIT w/ Cuboid	74.6	78.6	75.2	78.6
PRiFIT w/ Ellipsoid	75.4	78.7	75.3	79.0

Table 2: Comparison with state-of-the-art few-shot part segmentation methods on ShapeNet. Performance is evaluated using instance-averaged and class-averages IoU. † - We re-ran the publicly-released code from ACD [GRS*20] on our data splits, ensuring fair comparison.

posed by Wang *et al.* [WLF20] ‡. PRiFIT with ellipsoid as primitive outperforms previous methods including PRiFIT baseline with

‡ We run their code on all 16 categories on 5 different random splits

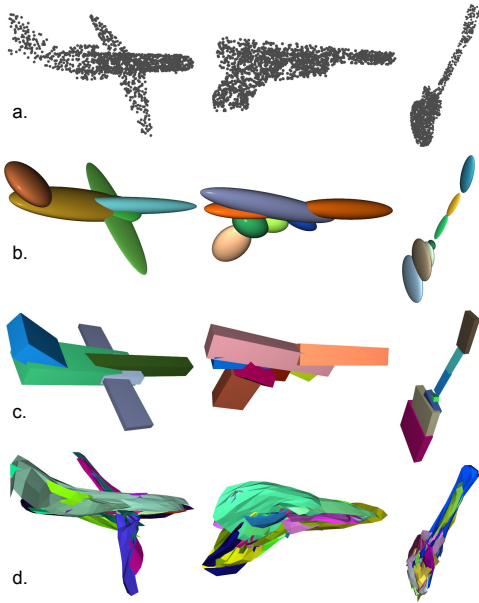


Figure 4: Visualization of various primitive fitting approaches. *a)* input point cloud. *b)* Ellipsoid fitting using our approach. *c)* cuboid fitting using our approach. *d)* different primitives from AtlasNet. Different colors are used to depict different primitives. For AtlasNet we visualize each chart with a unique color. Notice that geometric primitives are better localized and approximate the shape in fewer primitives in comparison to AtlasNet.

AtlasNet. Interestingly, PRiFIT with AtlasNet outperforms all previous baselines except ACD[§]. We speculate that since primitives predicted by AtlasNet are highly overlapped and less localized in comparison to our ellipsoid and cuboid fitting approaches as shown in Figure 4, this results in worse performance of AtlasNet.

PRiFIT with ellipsoid primitives gives the best results, outperforming ACD without having to rely on an external black-box method for the self-supervisory training signal. This shows that our approach of primitive fitting in an end-to-end trainable manner is better than training a network using contrastive learning guided by approximate convex decomposition of water-tight meshes as proposed by ACD.

In Figure 3 we show predicted semantic labels using PRiFIT along with fitted ellipsoids. In Figure 4 we show outputs of various self supervision techniques using primitive fitting. We experimented with both cuboid and ellipsoid fitting as a self supervision task. We observed that both performs similar qualitatively and quantitatively. The fitted primitives using ellipsoid/cuboid fitting approaches are more aligned with different parts of the shape in comparison to the outputs of AtlasNet.

Effect of the size of unlabeled dataset. In the Table 3 we show the effect of size of unlabeled dataset used for self-supervision. We

[§] We run their code on our random split for fair comparison.

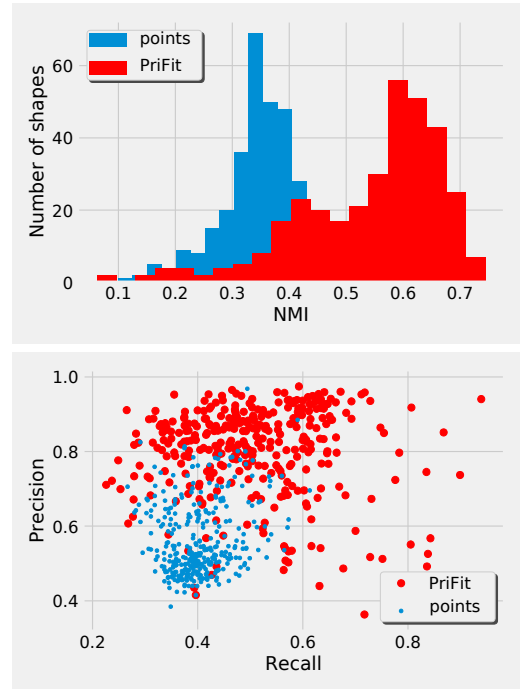


Figure 5: Analysis of clustering. We analyze two clustering approaches, 1) PRiFIT and 2) directly clustering points using K-Means. Top: normalized mutual information (NMI) and bottom: precision vs recall between predicted clusters and semantic part labels. PRiFIT gives higher average NMI (54.3 vs 35.4) and higher precision than clustering with only points as features.

size	0	2.5k	25k	52k
class avg. IOU	68.1	68.9	72.6	73.7

Table 3: Effect of the size of unlabeled dataset used for self-supervision on 5-shot semantic segmentation on ShapeNet.

observe improvement in performance of 5-shot semantic segmentation task with increase in unlabeled dataset.

Effect of similarity loss. Similarity loss is only used in the initial stage of training as it prevents the network converging to a local minima at this early phase. Without this procedure, our performance is similar to Baseline (training from scratch). However, using only similarity loss (without reconstruction loss) leads to worse results (44.2 mIOU) than Baseline (68.1 mIOU) on few-shot k=5 setting.

Analysis of learned point embeddings In Figure 5 we quantitatively analyze the performance of clustering induced by PRiFIT and compare it with clustering obtained by running the K-Means algorithm directly on point clouds. We take 340 shapes from the Airplane category of ShapeNet for this experiment and show the histogram of Normalized Mutual Information (NMI) [SG03] and the precision-recall curve [FM83] between predicted clusters and ground truth part labels. Our approach produces clusters with

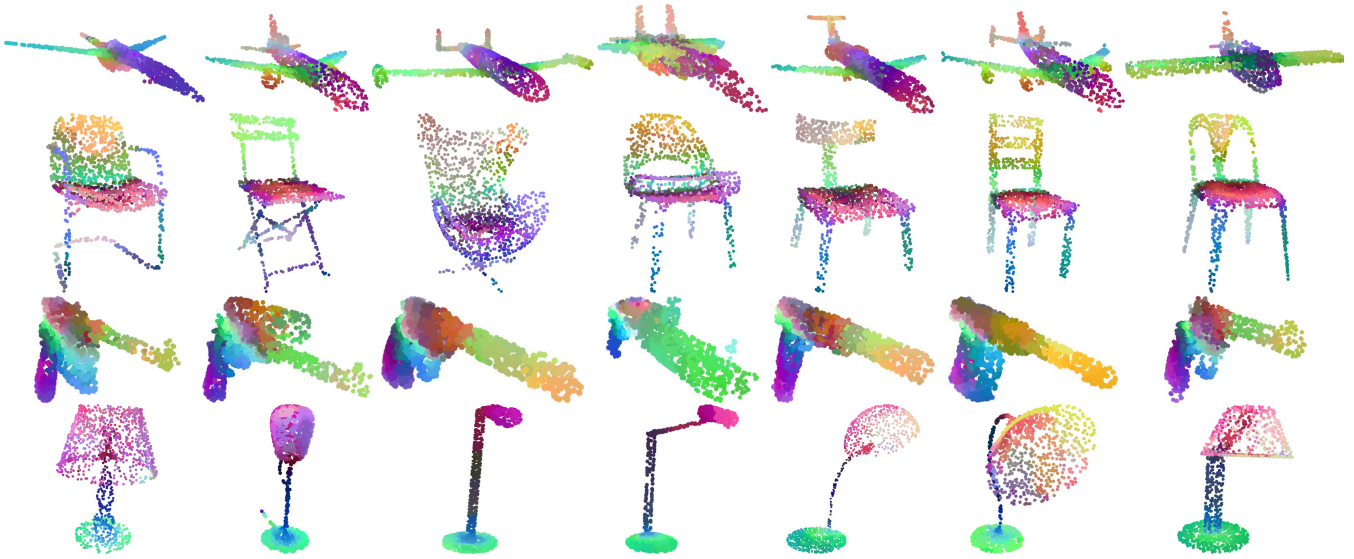


Figure 6: TSNE visualization of learned embeddings. For each shape category, we take a fixed number of shapes and extract point embeddings from PRiFIT. We run TSNE on each category separately to project the 128-D embeddings to 3D space. Notice that points belonging to same semantic parts are colored similarly, which indicates the consistency of learned embeddings.

higher NMI (54.3 vs 35.4), which shows better alignment of our predicted point clusters with the ground truth part labels. Our approach also produces higher precision clusters in comparison to K-Means at equivalent recall, which shows the tendency of our algorithm to over-segment a shape. To further analyze the consistency of learned point embedding across shapes, we use TSNE [vdMaa14] to visualize point embedding by projecting them to 3D color space. We use a fixed number of shapes for each category and run TSNE on each category separately. Figure 6 shows points belonging to same semantic parts are consistently projected to similar colors, further confirming the consistency of learned embeddings.

4.3. Few-shot semantic segmentation on PartNet

Here we experiment on the PartNet dataset for the task of few-shot semantic segmentation. For each category from this dataset, we randomly sample k labeled shapes and use these for training semantic segmentation part of the architecture. Similar to our previous experiment, we use the complete training shapes from the ShapeNet Core dataset for the self-supervision task. We choose DGCNN as a backbone architecture for this experiment. Unlike our ShapeNet experiments, in the PartNet experiment we train a separate model for each category, as done in the original paper [MZC*19b].

Similar to our previous experiment, we create two baselines – 1) we train a network from scratch providing only k labeled examples, and 2) we train the AtlasNet on the entire unlabeled training set using the self-supervised reconstruction loss and only k labeled examples as supervision.

Table 4 shows part avg. IOU for the different methods. The AtlasNet method shows improvement over the purely-supervised baseline. PRiFIT shows improvement over both AtlasNet and Baseline, indicating the effectiveness of our approach in the *fine-grained*

Samples/cls.	k=10	k=20	k=40
Baseline	27.2±0.7	31.6±0.6	36.7±0.9
PRiFIT w/ Atlas	28.5±0.7	31.7±0.7	36.5±0.7
PRiFIT w/ Cuboid	29.3±0.4	32.4±0.7	37.5±0.5
PRiFIT w/ Ellipsoid	29.4±0.7	32.6±0.6	37.6±0.4

Table 4: Few-shot segmentation on the PartNet dataset (part avg. IoU over 5 rounds). The number of shots or samples per class is denoted by k for each of the 12 PartNet categories used for supervised training. Our proposed method PRiFIT consistently outperforms the baseline.

semantic segmentation setting as well. We further experiment with cuboids as the primitive, which achieves similar performance as using ellipsoid primitives, consistent with our previous ShapeNet results.

Effect of intersection loss. We also experiment with adding intersection loss while training PRiFIT with ellipsoid primitives, as shown in Table 5. This gives improvements only on the PartNet dataset, and we speculate that since PartNet contains fine-grained segmentation of shapes, minimizing overlap between primitives here is more helpful than in the ShapeNet dataset, which contains only a coarse level of segmentation. On ShapeNet, we observed that the intersection loss is beneficial for 9 out of 16 categories (Airplane, Bag, Chair, Guitar, Lamp, Laptop, Mug, Rocket and Table). On the PartNet dataset we observed the intersection loss is beneficial for 8 out of 12 categories (Bed, Bottle, Dishwasher, Display, Knife, Microwave, StorageFurniture and TrashCan). In general, we observed that the benefits from this loss function varies with the

Method	ShapeNet		PartNet	
	cls. IoU		part avg. IoU	
	k=1%	k=5%	k=10	k=20
PRiFIT w/ Ellipsoids	75.3	79.0	28.9	32.1
PRiFIT w/ Ellipsoids+inter	75.0	79.0	29.4	32.6

Table 5: Effect of intersection loss. Average performance over all categories for PRiFIT trained with intersection loss (+inter) and without intersection loss on ShapeNet and PartNet dataset. PRiFIT trained with intersection loss (+inter) gives improvement on PartNet dataset.

category of shape and the coarseness of the segmentation. Similar observation was also made by Kawana et al. [KMH20].

5. Conclusion

We propose a simple semi-supervised learning approach, PRiFIT, for learning point embeddings for few shot semantic segmentation. Our approach learns to decompose an unlabeled point cloud into a set of geometric primitives, such as ellipsoids and cuboids, or alternatively deformed planes as in AtlasNet. We provide an end-to-end trainable framework for incorporating this task into standard network architectures for point cloud segmentation. PRiFIT can be readily applied to existing architectures for semantic segmentation and shows improvements over fully-supervised baselines and other approaches on the ShapeNet and PartNet datasets. This indicates that learning to reconstruct a shape using primitives can induce representations useful for discriminative downstream tasks. PRiFIT also has limitations: we currently rely on basic primitives of a single type (i.e. ellipsoid, cuboids, or deformed planes). Predicting combinations of primitives or other types of primitives for each shape could be useful to capture more part variability. In addition, our primitives capture the shape rather coarsely, making our representations less fit for fine-grained tasks. Finally, it would be interesting to explore unsupervised approaches based on primitive and other forms of surface fitting.

6. Acknowledgements

The work is supported in part by NSF grants 1908669 and 1749833. The experiments were performed using equipment obtained under a grant from the Collaborative Fund managed by the Mass. Tech. Collaborative.

References

[ABT20] ALLIEGRO, ANTONIO, BOSCAINI, DAVIDE, and TOMMASI, TATIANA. *Joint Supervised and Self-Supervised Learning for 3D Real-World Challenges*. 2020 3, 7.

[Bie87] BIEDERMAN, IRVING. “Recognition-by-components: a theory of human image understanding.” *Psychological review* 94.2 (1987) 2.

[Bin71] BINFORD, I. “Visual perception by computer”. *IEEE Conference of Systems and Control*. 1971 2.

[BLM87] BINFORD, THOMAS O, LEVITT, TOD S, and MANN, WALLACE B. “Bayesian inference in model-based machine vision”. *Proceedings of the Third Conference on Uncertainty in Artificial Intelligence*. 1987 2.

[CBC*01] CARR, JONATHAN C, BEATSON, RICHARD K, CHERRIE, JON B, et al. “Reconstruction and representation of 3D objects with radial basis functions”. *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*. 2001 2.

[CFG*15] CHANG, ANGEL X., FUNKHOUSER, THOMAS A., GUIBAS, LEONIDAS J., et al. “ShapeNet: An Information-Rich 3D Model Repository”. *CoRR abs/1512.03012* (2015) 1, 3, 6.

[CYF*19] CHEN, ZHIQIN, YIN, KANGXUE, FISHER, MATTHEW, et al. “BAE-NET: branched autoencoder for shape co-segmentation”. *Proceedings of the IEEE International Conference on Computer Vision*. 2019 2.

[DGY*20] DENG, BOYANG, GENOVA, KYLE, YAZDANI, SOROOSH, et al. “CvxNet: Learnable Convex Decomposition”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2020 2.

[FM83] FOWLKES, E. B. and MALLOWS, C. L. “A Method for Comparing Two Hierarchical Clusterings”. *Journal of the American Statistical Association* 78.383 (1983) 8.

[FP93] FANG, SHU-CHERNG and PUTHENPURA, SARAT. *Linear Optimization and Extensions: Theory and Algorithms*. USA: Prentice-Hall, Inc., 1993. ISBN: 0139152652 4, 12.

[GCV*19] GENOVA, KYLE, COLE, FORRESTER, VLASIC, DANIEL, et al. “Learning Shape Templates with Structured Implicit Functions”. *International Conference on Computer Vision*. 2019 2, 3.

[GFK*18] GROUEIX, THIBAUT, FISHER, MATTHEW, KIM, VLADIMIR G., et al. “AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation”. *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2018 1, 2, 4, 7.

[GGC*20] GADELHA, MATHEUS, GORI, GIORGIO, CEYLAN, DUYGU, et al. “Learning Generative Models of Shape Handles”. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020 3.

[GRS*20] GADELHA, MATHEUS, ROYCHOWDHURY, ARUNI, SHARMA, GOPAL, et al. “Label-Efficient Learning on Point Clouds using Approximate Convex Decompositions”. *European Conference on Computer Vision (ECCV)*. 2020 1, 3, 7.

[GSMC09] GAL, RAN, SORKINE, OLGA, MITRA, NILOY J., and COHEN-OR, DANIEL. “iWIRES: An Analyze-and-Edit Approach to Shape Manipulation”. *ACM Transactions on Graphics (Siggraph)* 28.3 (2009) 2.

[GSV*17] GORI, GIORGIO, SHEFFER, ALLA, VINING, NICHOLAS, et al. “FlowRep: Descriptive Curve Networks for Free-Form Design Shapes”. *ACM Transaction on Graphics* 36.4 (2017) 2.

[GWM18] GADELHA, MATHEUS, WANG, RUI, and MAJI, SUBHRANSU. “Multiresolution Tree Networks for 3D Point Cloud Processing”. *ECCV*. 2018 2.

[HH19] HASSANI, KAVEH and HALEY, MIKE. “Unsupervised multi-task feature learning on point clouds”. *Proceedings of the IEEE International Conference on Computer Vision*. 2019 2, 7.

[HR83] HOFFMAN, DONALD D and RICHARDS, WHITMAN. “Parts of recognition”. (1983) 2.

[IVS15] IONESCU, C., VANTZOS, O., and SMINCHISESCU, C. “Matrix Backpropagation for Deep Networks with Structured Layers”. *IEEE International Conference on Computer Vision (ICCV)*. 2015 6.

[KF18] KONG, SHU and FOWLKES, CHARLESS C. “Recurrent pixel embedding for instance grouping”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018 4.

[KMH20] KAWANA, YUKI, MUKUTA, YUSUKE, and HARADA, TATSUYA. “Neural Star Domain as Primitive Representation”. *Proceedings of the 34th International Conference on Neural Information Processing Systems*. 2020 10.

[KYB19] KAISER, ADRIEN, YBANEZ ZEPEDA, JOSE ALONSO, and BOUBEKEUR, TAMY. “A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data”. *Computer Graphics Forum* 38.1 (2019) 2.

- [Läu88] LÄUTER, H. “Silverman, B. W.: Density Estimation for Statistics and Data Analysis. Chapman & Hall, London – New York 1986, 175 pp.” *Biometrical Journal* 30.7 (1988) 4.
- [LCH18] LI, JIAXIN, CHEN, BEN M, and HEE LEE, GIM. “SO-Net: Self-organizing network for point cloud analysis”. *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2018 7.
- [LCL18] LI, JIAXIN, CHEN, BEN M, and LEE, GIM HEE. “SO-Net: Self-Organizing Network for Point Cloud Analysis”. *arXiv preprint arXiv:1803.04249* (2018) 2.
- [LGB*21] LIU, YANCHAQ, GUO, JIANWEI, BENES, BEDRICH, et al. “TreePartNet: Neural Decomposition of Point Clouds for 3D Tree Reconstruction”. *ACM Trans. Graph.* 40.6 (Dec. 2021) 3.
- [LMH*20] LUO, TIANGE, MO, KAICHUN, HUANG, ZHIAO, et al. “Learning to Group: A Bottom-Up Framework for 3D Part Discovery in Unseen Categories”. *arXiv preprint arXiv:2002.06478* (2020) 3.
- [LSD*18] LI, LINGXIAO, SUNG, MINHYUK, DUBROVINA, ANASTASIA, et al. *Supervised Fitting of Geometric Primitives to 3D Point Clouds*. 2018. eprint: arXiv:1811.08988 6.
- [MGV11] MACEDO, IVES, GOIS, JOAO PAULO, and VELHO, LUIZ. “Hermite radial basis functions implicit”. *Computer Graphics Forum*. Vol. 30. 1. Wiley Online Library. 2011 2.
- [MJ09] MARDIA, KANTI V and JUPP, PETER E. *Directional statistics*. Vol. 494. John Wiley & Sons, 2009 4.
- [MKC18] MURALIKRISHNAN, SANJEEV, KIM, VLADIMIR G., and CHAUDHURI, SIDDHARTHA. “Tags2Parts: Discovering Semantic Regions from Shape Tags”. *Proc. CVPR. IEEE*, 2018 2.
- [MN78] MARR, DAVID and NISHIHARA, HERBERT KEITH. “Representation and recognition of the spatial organization of three-dimensional shapes”. *Proceedings of the Royal Society of London. Series B. Biological Sciences* 200.1140 (1978) 2.
- [MN99] MAGNUS, JAN R. and NEUDECKER, HEINZ. *Matrix Differential Calculus with Applications in Statistics and Econometrics*. Second. John Wiley, 1999 6.
- [MON*19] MESCHEDER, LARS, OECHSLE, MICHAEL, NIEMEYER, MICHAEL, et al. “Occupancy Networks: Learning 3D Reconstruction in Function Space”. *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2019 2.
- [MZC*19a] MO, KAICHUN, ZHU, SHILIN, CHANG, ANGEL X., et al. “PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding”. *Computer Vision and Pattern Recognition (CVPR)*. 2019 1.
- [MZC*19b] MO, KAICHUN, ZHU, SHILIN, CHANG, ANGEL X., et al. “PartNet: A Large-Scale Benchmark for Fine-Grained and Hierarchical Part-Level 3D Object Understanding”. *Computer Vision and Pattern Recognition (CVPR)*. 2019 6, 9.
- [MZL*09] MEHRA, RAVISH, ZHOU, QINGNAN, LONG, JEREMY, et al. “Abstraction of Man-Made Shapes”. *ACM Transactions on Graphics* 28.5 (2009) 2.
- [PUG19] PASCHALIDOU, DESPOINA, ULUSOY, ALI OSMAN, and GEIGER, ANDREAS. “Superquadrics Revisited: Learning 3D Shape Parsing beyond Cuboids”. *Computer Vision and Pattern Recognition (CVPR)*. 2019 3.
- [PvGG20] PASCHALIDOU, DESPOINA, van GOOL, LUC, and GEIGER, ANDREAS. “Learning Unsupervised Hierarchical Part Decomposition of 3D Objects from a Single RGB Image”. *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*. 2020 3.
- [QSMG17] QI, CHARLES R, SU, HAO, MO, KAICHUN, and GUIBAS, LEONIDAS J. “PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation”. *Proc. CVPR*. 2017 4.
- [Qui] QUILEZ, INIGO. *Ellipsoid SDF*. <https://www.iquilezles.org/www/articles/ellipsoids/ellipsoids.htm>. Accessed: 2020-11-15 5.
- [QYSG17] QI, CHARLES R., YI, LI, SU, HAO, and GUIBAS, LEONIDAS. “PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space”. *Proc. NIPS*. 2017 1, 4, 7.
- [SG03] STREHL, ALEXANDER and GHOSH, JOYDEEP. “Cluster Ensembles — a Knowledge Reuse Framework for Combining Multiple Partitions”. *J. Mach. Learn. Res.* 3 (Mar. 2003) 8.
- [SHWA15] SCHULMAN, JOHN, HEESS, NICOLAS, WEBER, THEOPHANE, and ABBEEL, PIETER. “Gradient Estimation Using Stochastic Computation Graphs”. *Neural Information Processing Systems*. 2015 6.
- [SKM19] SHARMA, GOPAL, KALOGERAKIS, EVANGELOS, and MAJI, SUBHRANSU. “Learning Point Embeddings from Shape Repositories for Few-Shot Segmentation”. *International Conference on 3D Vision (3DV)*. 2019 2.
- [SLM*20] SHARMA, GOPAL, LIU, DIFAN, MAJI, SUBHRANSU, et al. “ParSeNet: A Parametric Surface Fitting Network for 3D Point Clouds”. *Computer Vision and Pattern Recognition (CVPR)*. 2020 6.
- [STD*21] SUN, WEIWEI, TAGLIASACCHI, ANDREA, DENG, BOYANG, et al. “Canonical Capsules: Self-Supervised Capsules in Canonical Pose”. *Advances in Neural Information Processing Systems*. Vol. 34. 2021 2.
- [SWK07] SCHNABEL, R., WAHL, R., and KLEIN, R. “Efficient RANSAC for Point-Cloud Shape Detection”. *Computer Graphics Forum* 26.2 (2007) 2.
- [TAG19] THABET, ALI, ALWASSEL, HUMAM, and GHANEM, BERNARD. “MortonNet: Self-Supervised Learning of Local Features in 3D Point Clouds”. *arXiv* (Mar. 2019). eprint: 1904.00230 2, 7.
- [TGB13] THIERY, JEAN-MARC, GUY, EMILIE, and BOUBEKEUR, TAMY. “Sphere-Meshes: Shape Approximation using Spherical Quadric Error Metrics”. *ACM Transaction on Graphics (Proc. SIGGRAPH Asia 2013)* 32.6 (2013), Art. No. 178 2.
- [TSG*17] TULSIANI, SHUBHAM, SU, HAO, GUIBAS, LEONIDAS J., et al. “Learning Shape Abstractions by Assembling Volumetric Primitives”. *Computer Vision and Pattern Recognition (CVPR)*. 2017 3.
- [vdMaa14] Van der MAATEN, LAURENS. “Accelerating t-SNE using Tree-Based Algorithms”. *Journal of Machine Learning Research* 15.93 (2014) 9.
- [WLF20] WANG, LINGJING, LI, XIANG, and FANG, YI. “Few-Shot Learning of Part-Specific Probability Space for 3D Shape Segmentation”. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2020 3, 7.
- [WSL*19] WANG, YUE, SUN, YONGBIN, LIU, ZIWEI, et al. “Dynamic Graph CNN for Learning on Point Clouds”. *ACM Transactions on Graphics (TOG)* 38.5 (2019) 1, 4.
- [XGG*20] XIE, SAINING, GU, JIATAO, GUO, DEMI, et al. “PointContrast: Unsupervised Pre-training for 3D Point Cloud Understanding”. *ECCV*. 2020 2.
- [XLG12] XIAN, CHUHUA, LIN, HONGWEI, and GAO, SHUMING. “Automatic cage generation by improved OBBs for mesh deformation”. *The Visual Computer* 28.1 (2012) 2.
- [YC21] YANG, KAIZHI and CHEN, XUEJIN. “Unsupervised Learning for Cuboid Shape Abstraction via Joint Segmentation from Point Clouds”. *ACM Trans. Graph.* 40.4 (July 2021) 2.
- [YFST18] YANG, YAOQING, FENG, CHEN, SHEN, YIRU, and TIAN, DONG. “Foldingnet: Point cloud auto-encoder via deep grid deformation”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018 2.
- [YHH*19] YANG, GUANDAO, HUANG, XUN, HAO, ZEKUN, et al. “Pointflow: 3d point cloud generation with continuous normalizing flows”. *Proceedings of the IEEE International Conference on Computer Vision*. 2019 2.
- [ZBDT19] ZHAO, YONGHENG, BIRDAL, TOLGA, DENG, HAOWEN, and TOMBARI, FEDERICO. “3D point capsule networks”. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2019 2, 7.

[ZYH*15] ZHOU, YANG, YIN, KANGXUE, HUANG, HUI, et al. “Generalized Cylinder Decomposition”. *ACM Trans. Graph.* 34.6 (2015) 2.

Appendix A: Supplementary Material

Additional training details For few-shot experiments on Shapenet dataset, we train PointNet++ architecture on 4 Nvidia 1080Ti GPUs. We use batch size of 24 for all methods. The K-shot baseline is trained for approximately $K \times 200$ iterations while PRIFIT is always trained with approximately $K \times 1000$ iterations. The average time to predict primitives for a single shape is 98 ms on a single Nvidia 1080Ti GPU.

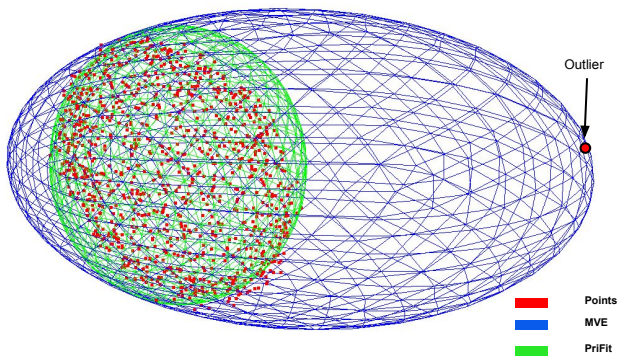


Figure 7: Robustness to outliers. An example of outlier-robust fitting with our method in contrast to MVE (minimum volume ellipsoid) that is sensitive to outliers. Our fitting result shown in green closely fits the input points (red) while ignoring the outlier, whereas MVE approach (blue) is sensitive to the outlier.

Robustness of ellipsoid fitting. Fig. 7 shows the robustness of our approach to outliers in comparison to minimum volume ellipsoid (MVE) [FP93]. Our approach takes into account the membership of the point to a cluster. For this example, we use a simple membership function $1/r^4$, where r is the distance of point from the center of the cluster and incorporates these per-point weights to estimate the parameters of ellipsoid in a closed form using SVD.