Unsupervised 3D Shape Parsing with Primitive Correspondence

Tianshu Zhao Yanran Guan Oliver van Kaick†

School of Computer Science, Carleton University, Canada

Abstract

3D shape parsing, the process of analyzing and breaking down a 3D shape into components or parts, has become an important task in computer graphics and vision. Approaches for shape parsing include segmentation and approximation methods. Approximation methods often represent shapes with a set of primitives fit to the shapes, such as cuboids, cylinders, or superquadrics. However, existing approximation methods typically rely on a large number of initial primitives and aim to maximize their coverage of the target shape, without accounting for correspondences among the primitives. In this paper, we introduce a novel 3D shape approximation method that integrates reconstruction and correspondence into a single objective, providing approximations that are consistent across the input set of shapes. Our method is unsupervised but also supports supervised learning. Experimental results demonstrate that integrating correspondences into the fitting process not only provides consistent correspondences across a set of input shapes, but also improves approximation quality when using a small number of primitives. Moreover, although correspondences are estimated in an unsupervised manner, our method effectively leverages this knowledge, leading to improved approximations.

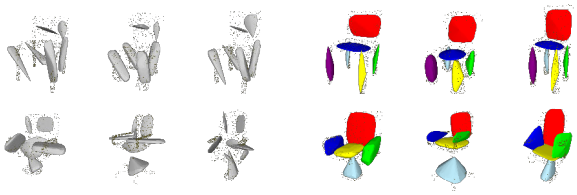
CCS Concepts

• ***Computing methodologies*** → ***Shape modeling;***

1. Introduction

Humans naturally decompose objects into parts to make sense of their structure, function, and purpose—whether recognizing a chair by its seat and legs or assembling a product by understanding its components [Bie85]. Thus, research in computer vision and graphics has also focused on analyzing the part composition of 3D shapes, which can be regarded as *shape parsing*. One line of work introduced methods for shape segmentation, where the goal is to decompose an input polygonal mesh or point cloud into meaningful parts, possibly based on example segmentations given as training data. For example, recent shape segmentation methods use supervised learning based on neural networks such as convolutional neural networks (CNNs) [YSGG17, KAMC17, WGS*18], autoencoders [CCZZ24], transformers [LASM24, LHJ*22], or graph networks [Roy23, SSSS21].

Another line of work related to part analysis seeks to approximate a shape with a set of primitive shapes. Representing a shape as a set of primitives provides a higher-level abstraction [SZTL19] of the shape, and more easily enables applications such as primitive-based editing [TSG*17]. Different types of primitives have been used for approximating shapes, such as cuboids [TSG*17, SZTL19], cylinders [Bin71], superquadrics [Pen86, PUG19, LWRC23], and structured implicit functions [GCV*19, GCS*20]. These works focus on reconstructing input shapes with primitives and thus provide good approximations of the shapes.



(a) *Without correspondence* **(b)** *With correspondence*

Figure 1: *3D shape parsing with and without primitive correspondence. Each colored shape represents a primitive and the color denotes the primitive's correspondence across the set of shapes. The point clouds being fit are shown in gray. We compare the shape parsing performance between a method targeting purely reconstruction (on the left) and our method which also establishes correspondences among the primitives (on the right). The results show that learning correspondence during the fitting can lead to a more accurate shape approximation in an unsupervised approach.*

However, none of the existing works cited above consider establishing a correspondence between the primitives. Specifically, the approximation methods rely on a large number of primitives

† Corresponding author

and aim to maximize the coverage of the target shape's surface, without accounting for correspondences between primitives of different shapes. A correspondence between primitives is desirable in certain contexts, such as when we require a consistent abstraction for all the shapes in a set. Primitives with correspondence can also enable applications such as the learning of generative models, construction of templates and statistical shape models of primitives, or collection editing, including collective part editing, deformation, exchange, and deletion.

Thus, in this paper, we introduce an unsupervised method for approximating a set of input shapes with primitives, so that the primitives approximate well the input and at the same time possess a correspondence across the set (Figure 1). We focus on fitting shapes with superquadric primitives, since superquadrics are more expressive than other representations by capturing a diverse class of shapes in a single parameter space [PUG19], such as cuboids and ellipsoids. Specifically, we extend existing unsupervised fitting methods based on a reconstruction loss by introducing a soft correspondence loss. The novel correspondence loss is guided by shape features extracred by a pre-trained, unsupervised foundation model, which allows us to compute the probability of primitive matches. Once the approximation is computed, we can also derive a consistent segmentation of all the shapes in the set if desired.

One alternative to fitting primitives with correspondence is to first segment the shapes and then fit primitives to the segments. However, this approach leads to less accurate fitting, since errors in the segmentation are propagated to the fitting step and cannot be corrected based on the collective correspondence. The method of Yang and Chen [YC21] is related to this type of approach, although it simultaneously computes a shape abstraction into cuboids and co-segmentation of a set of shapes. However, since their network is based on an encoding-decoding model, enough data is needed for training the feature encoder. In contrast, our method can also be applied to small sets of shapes since no encoding is necessary. A second alternative to our method is to define a template and fit it to all the shapes [KLM*13]. However, this requires constructing the template beforehand and does not benefit from taking all the shapes into account during fitting.

In contrast to the alternatives described above, our contributions can be summarized as follows:

- We introduce a method to approximate a set of input shapes with superquadric primitives while also establishing a correspondence between the primitives. Our method is capable of fitting primitives in a supervised or unsupervised manner.
- For unsupervised approximation, we introduce a novel correspondence loss as part of the combined loss for our deep network to learn the approximation, which captures point-to-point correspondences.
- We demonstrate with experimental validation that our method enables parsing of 3D shapes by obtaining approximations with good quality and consistent primitive correspondence. We trained our model with datasets of varying quality (e.g., poor correspondence and small datasets). The results (Figure 6 and Table 1) show the surprising performance of our method, especially on small datasets, demonstrating its flexibility and robustness.
- Additionally, we design a metric called penalized bi-directional Rand index (PBRI) to evaluate the correspondence of unsupervised shape parsing approaches. This index can work with the probability of existence and does not require one-to-one labels between points on the primitives and the input point cloud.

Our implementation is available at https://github.com/YujinK-CHN/USPC.

2. Background and Related Work

In this section, we briefly review the technical background and previous work related to our paper.

2.1. Superquadric Parametrization

Superquadrics provide a vocabulary for shape representation, often used for modeling real-world objects as compositions of simple parts. Pentland [Pen86] is among the earliest to use superquadrics for shape parsing aligned with our recognition of "parts" in scene structures. A superquadric can be described by the following parametric equation [PUG19]:

$$\mathbf{r} = \begin{bmatrix} \alpha_1 \cos^{\epsilon_1}\eta \cos^{\epsilon_2}\omega \\ \alpha_2 \cos^{\epsilon_1}\eta \sin^{\epsilon_2}\omega \\ \alpha_3 \sin^{\epsilon_1}\eta \end{bmatrix}, \tag{1}$$

where $\mathbf{r}$ is a 3D vector that sweeps out a surface parameterized by latitude η and longitude ω, with $-\frac{\pi}{2} \leq \eta \leq \frac{\pi}{2}$ and $-\pi \leq \omega \leq \pi$, $\alpha = (\alpha_1, \alpha_2, \alpha_3)$ determines the size of the shape along each dimension, and $\epsilon = (\epsilon_1, \epsilon_2)$ determines the global shape of the superquadric, making it "rounder" or more "squarish". Superquadrics can describe primitive shapes such as spheres, ellipsoids, cylinders, and smooth cubes and diamonds. This representation can be extended by applying a translation $\mathbf{t}$ and rotation $\mathbf{q}$ to describe a superquadric in any orientation and position in space. Thus, such a general superquadric can be represented with 11 parameters from α, ϵ, $\mathbf{t}$, and $\mathbf{q}$, assuming a 3D translation vector $\mathbf{t}$ and normalized quaternion $\mathbf{q}$.

2.2. Learning-Based Shape Parsing

For shape parsing, Zou et al. [ZYY*17] introduced 3D-PRNN, a generative recurrent neural network using long-short term memory networks and mixture density networks to encode symmetry in depth images, enabling cuboid primitive decoding. Niu et al. [NLX18] proposed a convolutional-recursive autoencoder that extracts shape structures from RGB images and recursively decodes cuboid hierarchies while modeling part relations. Tulsiani et al. [TSG*17] developed an unsupervised CNN-based method that predicts volumetric primitives by minimizing the discrepancy between input meshes and reconstructed assemblies. Paschalidou et al. [PUG19] extended this by leveraging superquadric representations and demonstrating the tractability of bi-directional chamfer distance as a loss function. Because superquadrics offer a flexible, compact, and differentiable way to represent geometric primitives, more 3D shape parsing and primitive fitting methods have been developed using them. For instance, Liu et al. [LWRC22] present a probabilistic optimization framework to robustly recover superquadric parameters from 3D point clouds.

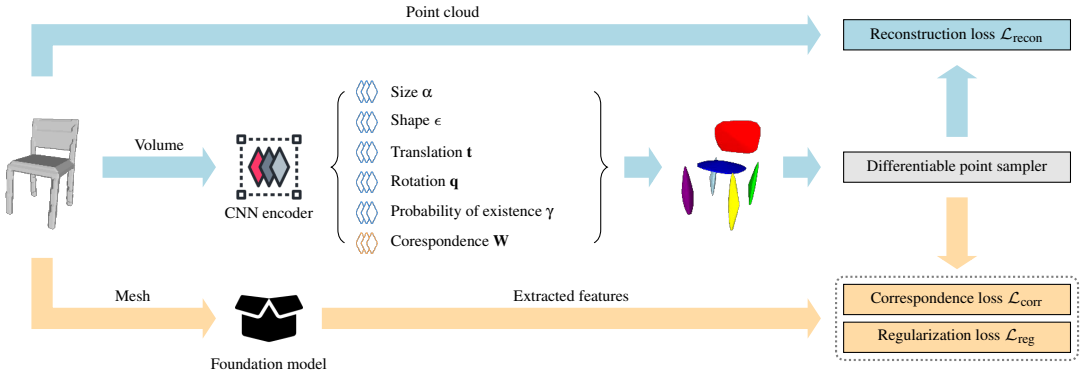


Figure 2: *Overview of our shape parsing method. Given an input shape represented as a mesh, volume, and point cloud, the network at the center encodes the volume with a CNN which forwards the encoding to multi-head layers that predict superquadric parameters. The blue trajectory represents the reconstruction process, while the yellow trajectory represents the process of learning the correspondence.*

Alaniz et al. [AMA23] propose an iterative method to reconstruct 3D objects by fitting and recomposing superquadric primitives from multi-view observations. In addition, for man-made objects such as computer-aided design models, methods have been developed to fit geometric primitives in forms other than superquadrics, such as planes, spheres, cylinders, and cones, for shape segmentation or reconstruction [LSD*19, YYM*21, LSC*21, LLY*23].

Yang et al. [YC21] is the most similar work to ours, predicting parameterized cuboids for point cloud segmentation. The key differences to our work include: (i) our approach fits superquadrics, predicting more parameters; (ii) we introduce a novel correspondence loss integrated into training, rather than learning segmentation and approximation with separate network branches; (iii) our approach does not require learning a feature encoder from the data, instead employing a foundation model.

2.3. Correspondence and Segmentation

Recent work in 3D shape modeling has focused on establishing correspondences and performing segmentation within or across shape categories.

Co-segmentation methods, in particular, leverage part correspondences across shapes within a category to produce consistent segmentations. For example, Zhu et al. [ZXC*20] propose a weakly supervised method that adaptively co-segments 3D shapes by learning consistent part correspondences using a group consistency loss. Chen et al. [CYF*19] introduce BAE-NET, an unsupervised branched autoencoder to co-segment 3D shapes by learning recurring parts through reconstruction.

With recent advances in foundation models, new methods address correspondence and segmentation beyond category boundaries. Dutt et al. [DMM24] propose Diff3F, a feature distiller that extracts semantic features by rendering shapes from multiple views and generating depth/normal maps. These maps guide ControlNet-based [ZRA23] photo-realistic image synthesis via stable diffusion [RBL*22], with the diffusion features then aggregated onto input surfaces to enable semantic correspondence without additional training. Diff3F leverages foundation models for vision to extract semantic descriptors in a zero-shot manner, employing DINOv2 [ODM*24] to learn robust visual features from large, unlabeled datasets. While Diff3F produces high-level semantic features, we seek instance segmentation for correspondence learning. Notably, Diff3F's semantic features remain distinguishable within the same object for functionally similar components (e.g., legs). See Figure 3 for examples of the feature similarity.

3. Shape Parsing with Correspondence

Our method is based on the work of Paschalidou et al. [PUG19], where we perform the primitive fitting with a neural network trained in an iterative optimization. However, we do not use all terms of their loss function, and instead incorporate the estimation of correspondences into the fitting process, which requires the design of a new soft correspondence loss. The method operates in an unsupervised manner, although it can also be used with supervision. In each iteration, the network predicts a set of primitives for an input shape, and then corrects the prediction based on a loss function incorporating reconstruction and correspondence objectives. At the end of the optimization, the network will have learned to perform an accurate prediction of primitives for an input shape. As Paschalidou et al. [PUG19] and Tulsiani et al. [TSG*17], we represent primitives as superquadrics. As follows, we first provide a general overview of the network. Then, we discuss each term of the loss function used in the optimization. A diagram providing an overview of our method is shown in Figure 2.

Input and output. Our method takes as input a set of shapes and computes an approximation of each shape with a set of primitives in correspondence across the set. Each shape is given as a mesh. We extract an occupancy grid (volume) and a point cloud X from the shape. We use the mesh for feature extraction, the volume for feature encoding, and the point cloud for chamfer distance computation. We also take as input the maximum number of initial primitives M, and the expected number of primitives k per shape. The output reconstruction is not constrained to k primitives. However, k provides us with an initial guess on the number of labels in a shape, and is fixed throughout the optimization. Our method outputs a set

of up to M primitives for each input shape, which are ordered consistently according to their correspondence across the set.

Network architecture. As in the work of Paschalidou et al. [PUG19], our architecture is a multi-head neural network that predicts a set of up to M primitives that approximate the input shape. The network consists of a 3D CNN encoder followed by five independent heads that predict each parameter of a superquadric primitive: size α, shape ϵ, translation $\mathbf{t}$, rotation $\mathbf{q}$, and probability of existence γ. We add an independent head for estimating a predicted correspondence matrix $\mathbf{W} \in \mathbb{R}^{M \times k}$ as a differentiable probability distribution, where k is the expected number of primitives.

3.1. Loss Function

Our network is optimized with the loss $\mathcal{L}$:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{\text{recon}} + \lambda_2 \mathcal{L}_{\text{corr}} + \lambda_3 \mathcal{L}_{\text{reg}}, \tag{2}$$

where $\mathcal{L}_{\text{recon}}$ is the reconstruction loss, $\mathcal{L}_{\text{corr}}$ is the correspondence loss, $\mathcal{L}_{\text{reg}}$ is a cosine similarity regularization loss for the predicted correspondence, and λ_1, λ_2, and λ_3 are the coefficients for balancing the loss components.

3.1.1. Reconstruction Loss

The approximation loss is given by a bi-directional chamfer distance computation as in the work of Paschalidou et al. [PUG19]:

$$\mathcal{L}_{\text{recon}} = \mathcal{L}_{\mathcal{Y} \to X} + \mathcal{L}_{X \to \mathcal{Y}}, \tag{3}$$

where X indicates the input point cloud containing N points, i.e., $X = \{\mathbf{x}_i \mid i \in \{1, \dots, N\}\}$, and $\mathcal{Y}$ denotes the set of predicted primitives, where each primitive is represented as a set of S points sampled from its surface, e.g., the mth primitive Y_m is represented as $\{\mathbf{y}_j^m \mid j \in \{1, \dots, S\}\}$. $\mathcal{L}_{\mathcal{Y} \to X}$ represents a primitive-to-point cloud distance, which computes the average minimum distance from each point in each primitive of $\mathcal{Y}$ to its nearest neighbor in X:

$$\mathcal{L}_{\mathcal{Y} \to X} = \sum_{m=1}^{M} \gamma_m \mathcal{L}_{Y_m \to X}, \tag{4}$$

with

$$\mathcal{L}_{Y_m \to X} = \frac{1}{S} \sum_{j=1}^{S} \min_{i \in \{1, \dots, N\}} \|\tau_m(\mathbf{x}_i) - \mathbf{y}_j^m\|, \tag{5}$$

where γ_m is the probability of existence of the mth primitive, and $\tau_m(\cdot)$ is a transformation function that transforms a 3D point in world coordinates into the local coordinate system of the mth primitive. More details on $\tau_m(\cdot)$ can be found in [PUG19].

The point cloud-to-primitive direction is mainly used to ensure that the target is covered by the predicted primitives. Thus, in $\mathcal{L}_{X \to \mathcal{Y}}$, we measure only the distance from existing primitives. Therefore, the point cloud-to-primitive loss can be defined as:

$$\mathcal{L}_{X \to \mathcal{Y}} = \sum_{i=1}^{N} \sum_{m=1}^{M} \Delta_i^m \gamma_m \prod_{\bar{m}=1}^{m-1} (1 - \gamma_{\bar{m}}), \tag{6}$$

where

$$\Delta_i^m = \min_{j \in \{1, \dots, S\}} \|\tau_m(\mathbf{x}_i) - \mathbf{y}_j^m\|. \tag{7}$$

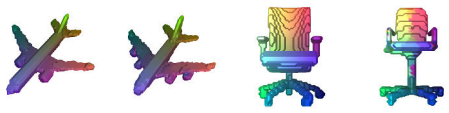


Figure 3: *Color maps of features extracted with the foundation model Diff3F [DMM24], based on cosine similarity between a pair.*

Paschalidou et al. [PUG19] provide the full details of this reconstruction loss, which we use in our method.

3.1.2. Feature Extraction and Clustering

Our correspondence loss is based on the similarity of point features. Thus, we utilize a method based on a foundation model [DMM24] as a black box for feature extraction. The foundation model takes an entire mesh of a shape as input and generates a feature vector for each mesh vertex. For each point in the input point cloud, we assign a feature based on the feature vector of its nearest mesh vertex, resulting in a sequence of N features vectors $\{\mathbf{f}_i \mid i \in \{1, \dots, N\}\}$.

Next, we assume that the input is composed of the expected number of labels k and thus we group the points into k clusters based on their feature vectors. We assume that the clusters represent primitives that likely compose the input, which we call the "reference primitives". We leverage k-means clustering for this step. Specifically, we first derive k cluster centers $\{\mathbf{c}_j \mid j \in \{1, \dots, k\}\}$ from the feature vectors. Then, we derive a reference correspondence matrix $\mathbf{P} = (p_{i,j})_{N \times k}$, with each entry $p_{i,j}$ computed as:

$$p_{i,j} = \text{softmax}(\|\mathbf{f}_i - \mathbf{c}_j\|), \tag{8}$$

where $p_{i,j}$ can be interpreted as the probability of the ith point corresponding to the jth reference primitive, according to the extracted features. Since we define $\mathbf{P}$ as a probability distribution, we use the softmax function to ensure that all entries lie within $[0, 1]$. Note that $\mathbf{P}$ is only computed once at the beginning of the optimization.

3.1.3. Correspondence Loss

The main idea of the correspondence loss is to match the input points to points sampled on the predicted primitives based on the features extracted in Section 3.1.2. However, to simplify the formulation, we pose the problem as matching the correspondence vectors of input points to the vectors of points sampled from the predicted primitives. The vectors for input points encode the probability of a point matching to one of k hypothetical reference primitives, while the vectors for a predicted point encode its probability of matching to one of the predicted primitives. Thus, the problem is reduced to matching the reference to the predicted primitives. Moreover, since features around a primitive may be similar, we also consider the spatial distance between points in the matching. In this setting, we are given the probabilistic correspondence matrix $\mathbf{P}$ from the foundation model, while the network also predicts a probability matrix $\mathbf{W}$. Note that the network will learn to perform this prediction during the optimization. Then, we match the points based on these two matrices and their spatial distances.

Specifically, we denote $\mathbf{p}_i = (p_{i,1}, \dots, p_{i,k})$ as the ith row of $\mathbf{P}$, representing the reference correspondence vector for the ith point in

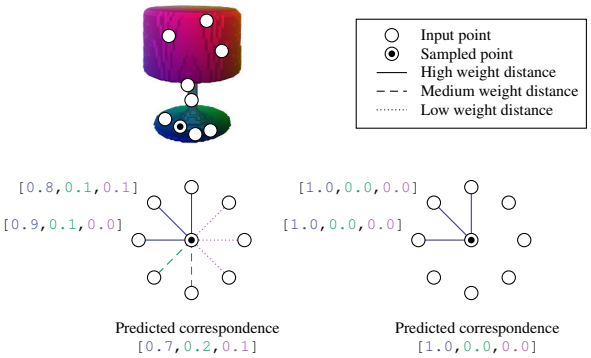


Figure 4: *Illustration of the correspondence vector for a point sampled from a primitive (at the center) surrounded by input points, with the best matches indicated by solid lines. Left: using feature vectors. Right: using pseudo-labels.*

X, where $i \in \{1,\ldots,N\}$. We denote $\mathbf{w}_m = (w_{m,1},\ldots,w_{m,k})$ as the mth row of $\mathbf{W}$, representing the predicted correspondence vector for the mth primitive. Given each point $\mathbf{y}_j^m$ ($j \in \{1,\ldots,S\}$) on the mth primitive, we calculate a distance vector $\mathbf{d}_j^m = (d_{j,1}^m,\ldots,d_{j,N}^m)$ between $\mathbf{y}_j^m$ to each point in X. Thus, we define the correspondence loss on each sample point $\mathbf{y}_j^m$ as:

$$\mathcal{L}_{\mathbf{y}_j^m} = \frac{1}{N}\sum_{i=1}^{N} d_{j,i}^m \max(\mathbf{p}_i \odot \mathbf{w}_m). \tag{9}$$

Therefore, the correspondence loss on the mth primitive Y_m is defined as:

$$\mathcal{L}_{Y_m} = \frac{1}{S}\sum_{j=1}^{S} \mathcal{L}_{\mathbf{y}_j^m}. \tag{10}$$

We sum up the correspondence loss on all the primitives weighted by their probabilities of existence to derive the overall correspondence loss for the entire shape, i.e.,

$$\mathcal{L}_{\text{corr}} = \sum_{m=1}^{M} \gamma_m \mathcal{L}_{Y_m}. \tag{11}$$

According to (9), as the predicted correspondence gets close to the correspondence vector of a reference point and concentrates on the same region, its corresponding distance contributes more to the loss, which causes the model to pay more attention to reducing the distance between points and primitives. Thus, the primitive is pulled towards the corresponding area.

We illustrate the calculation of the $\mathcal{L}_{\mathbf{y}_j^m}$ term of the loss in the left column of Figure 4, where the reference correspondence vectors are derived from the features of the foundation model. The illustration shows the correspondence vectors for a predicted point and multiple input points that it can match to, while the matches with solid lines have higher weighted distances.

3.1.4. Variations of our method

The right column of Figure 4 shows a variation of our unsupervised method where we encode the correspondences as one-hot-encoded labels, which we call *pseudo-labels* since the labels are provided by the segmentation given by the foundation model. Finally, we can also train our method in a supervised manner by one-hot encoding ground truth (GT) labels, if these are available.

3.1.5. Correspondence Regularization Loss

We introduce a third term to the loss function which serves two purposes: (i) it prevents multiple primitives from mapping to the same region of a shape, ensuring that the primitives spread out to cover the shape rather than concentrating on the same regions, and (ii) it maintains the correspondences fixed once primitives locked to different regions, allowing only the reconstruction fit to be improved.

The term is defined as the differentiable cosine similarity with existence probabilities for the set of primitives:

$$\mathcal{L}_{\text{reg}} = \frac{\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M} \gamma_i\gamma_j\left(1-\cos(\theta_{i,j})\right)}{\sum_{i=1}^{M}\sum_{j=1,j\neq i}^{M} \gamma_i\gamma_j}, \tag{12}$$

with

$$\theta_{i,j} = \frac{\mathbf{w}_i \cdot \mathbf{w}_j}{\|\mathbf{w_i}\|\|\mathbf{w_j}\|}, \tag{13}$$

where γ_i and γ_j are the probabilities of existence of the ith and jth primitives, and $\mathbf{w}_i$ and $\mathbf{w}_j$ represent the predicted correspondence vectors of the ith and jth primitives.

This loss penalizes the similarity between the correspondence of primitives and thus encourages the network to explore a better approximation.

3.2. Schedule for Dynamic Loss Balancing

We randomly initialize the weights of the network and optimize it with the Adam optimizer. In order to avoid bad local minima in our optimization, which translate into suboptimal primitive fittings, we use a scheduling approach for dynamically weighting the terms of the loss function. Specifically, we divide the optimization iterations into three phases, which are illustrated in the top row of Figure 9. In the first phase starting at the first iteration, we set a larger weight for the correspondence regularization loss, which encourages primitives to distribute themselves around the shape and avoid overlaps. Next, we increase the weight of the correspondence loss, which leads the primitives to position themselves in specific areas of the shapes which most likely reduce the correspondence loss. This is exemplified in the 20th iteration of Figure 9. Finally, we increase the weight of the reconstruction loss, so that the primitives better fit and reconstruct the shape, and so that imperfect correspondence features have less effect in the final results. This occurs between the 40th and 300th iterations in Figure 9. Note that, for optimal results in the unsupervised setting, the range of the three phases may need to be experimentally adjusted.

4. Experimental Evaluation

In this section, we present the experimental setup for evaluating our model's 3D shape parsing performance and provide evidence that our network can learn and leverage correspondence information within a single objective during unsupervised shape approximation training.

We evaluate our model under supervised and unsupervised settings. Both supervised and unsupervised models use the same formulation to compute $\mathcal{L}_{\text{corr}}$. However, the supervised method uses GT labels derived from the GT segmentations given with the datasets, while the unsupervised setting uses pseudo-labels or features provided by the foundation model, as discussed at the end of Section 3.1.3. Moreover, to the best of our knowledge, there is no other method that performs superquadric fitting while establishing primitive correspondence. Thus, we use the method of Paschalidou et al. [PUG19] as the main baseline for comparison. We run the method with all the losses described in their paper, including a parsimony loss that reduces the number of primitives. As an additional baseline, we also evaluate an alternative to our method, i.e., segmenting the shapes first with the foundation model and then fitting superquadrics to the segments via principal component analysis; we refer to this as "seg + fit" for brevity. Lastly, we compare with the method of Yang and Chen [YC21] which learns shape abstractions using cuboids as primitives in an unsupervised manner by simultaneously segmenting the input meshes.

4.1. Datasets and Implementation Details

We use the ShapeNet [CFG*15] and PartNet [MZC*19] datasets of 3D shapes in our experiments. We focused on the *airplane* and *car* categories from ShapeNet, and the *chair* and *lamp* categories from PartNet, given that these categories have complex part structures. We primarily use the raw shapes from both datasets, with part labels from PartNet used in the experiments with supervised learning. The training set consists of 126 regular chairs, 67 office chairs, 84 lamps, 103 airplanes, and 207 cars, while the test set includes 10 shapes from each category. The shapes in these datasets are consistently aligned. However, note that our method does not require pre-alignment since the features of the foundation model are rotation-invariant.

We set the initial number of primitives to a maximum of $M = 8$, with each category having an expected number of primitives k (where k applies only to our model). The training process runs for 1000 epochs with a learning rate of 10^{-4}. To compute the loss, 2048 points are uniformly sampled from the mesh's surface, and 256 points are sampled from each primitive during every iteration.

4.2. Evaluation Metrics

To evaluate the reconstruction quality of the results, we measure the chamfer distance between the point cloud and the union of points sampled from all primitives generated.

To evaluate the quality of the fitting with correspondences, we use the Rand Index measure, which is commonly used for evaluating segmentation results [CGF09]. We use the measure to compare each primitive to the GT labels of the points and estimate how well the primitives fit to the semantics parts of the shape. We can compute the Rand index per point. However, since we randomly sample points on the superquadrics, the sampling density may not be uniform across the shape. Thus, direct pairwise comparison of points may not be effective. Instead, we propose a modified Rand index, PBRI, which compares primitives to semantic parts also based on spatial proximity rather than just label matching. Points are considered correctly clustered if they are within a distance threshold from the corresponding semantic part in the GT. Our PBRI ranges from 0 (no correspondence) to 1 (perfect correspondence), reflecting clustering accuracy based on labels and spatial proximity. We briefly describe the computation of our PBRI as follows.

Given two sets of points $A = \{\mathbf{a}_i \mid i \in \{1,\dots,|A|\}\}$ and $B = \{\mathbf{b}_i \mid i \in \{1,\dots,|B|\}\}$, we define a Spatially-aware, Probability-weighted Rand Index (SPRI) from A to B, computed as:

$$\text{SPRI}(A,B) = \frac{\sum_{i=1}^{|A|}\sum_{j=i+1}^{|A|}\sigma_{i,j}\mathbb{1}_{\text{TP}}(\mathbf{a}_i,\mathbf{a}_j \mid B) + \sigma_{i,j}\mathbb{1}_{\text{TN}}(\mathbf{a}_i,\mathbf{a}_j \mid B)}{\binom{|A|}{2}}, \tag{14}$$

where the indicator $\mathbb{1}_{\text{TP}}(\mathbf{a}_i,\mathbf{a}_j \mid B)$ is 1 when the point pair $(\mathbf{a}_i,\mathbf{a}_j)$ is classified as a true positive w.r.t. B, i.e., (i) the two points are spatially close to their closest points in B (within a predefined distance threshold), (ii) they have the same label, and (iii) their closest points in B have the same label. Similarly, $\mathbb{1}_{\text{TN}}(\mathbf{a}_i,\mathbf{a}_j \mid B)$ is 1 when the point pair $(\mathbf{a}_i,\mathbf{a}_j)$ is classified as a true negative w.r.t. B, i.e., (i) the two points are spatially close to their closest points in B (within a predefined distance threshold), (ii) they have different labels, and (iii) their closest points in B have the different labels. The term $\sigma_{i,j}$ represents the average of γ_i and γ_j, which are the probabilities of existence of the closest points to $\mathbf{a}_i$ and $\mathbf{a}_j$ in B. Note that, if B contains points from the GT point cloud, all the probabilities of existence are set to 1; if B contains points sampled on the predicted primitives, the probability of existence of each point is that of the predicted primitive it belongs to.

Then, we first define the labels associated with each point as follows. For the input point cloud X, the label of each point $\mathbf{x}_i$ ($i \in \{1,\dots,N\}$) is defined as:

$$l_{\text{GT}}(\mathbf{x}_i) = \arg\max_j(\{p_{i,j} \mid j \in \{1,\dots,k\}\}). \tag{15}$$

For the points on the predicted primitives, e.g., a point $\mathbf{y}_i^m$ ($i \in \{1,\dots,S\}$) on the mth primitive, its label is defined as:

$$l_{\text{pred}}(\mathbf{y}_i^m) = \arg\max_j(\{w_{m,j} \mid j \in \{1,\dots,k\}\}). \tag{16}$$

Furthermore, given the set of labels of the points in the GT point cloud as $L_{\text{GT}} = \{l_{\text{GT}}(\mathbf{x}_i) \mid i \in \{1,\dots,N\}\}$ and the labels of the points on the predicted primitives as $L_{\text{pred}} = \{l_{\text{pred}}(\mathbf{y}_i^m) \mid i \in \{1\dots,S\}, m \in \{1,\dots,M\}\}$, we define the coverage penalty Q as:

$$Q = \frac{|\text{unique}(L_{\text{GT}}) \setminus \text{unique}(L_{\text{pred}})|}{k}, \tag{17}$$

where unique$(\cdot)$ is a function that returns the unique elements of the input set.

Finally, based on (14) and (17), we define the PBRI between the GT point cloud X and the point set $Y = \bigcup_{i=m}^{M} Y_m$ containing the points from all predicted primitives as:

$$\text{PBRI}(X,Y) = \max\left(\frac{\text{SPRI}(X,Y) + \text{SPRI}(Y,X)}{2} - Q, 0\right). \tag{18}$$

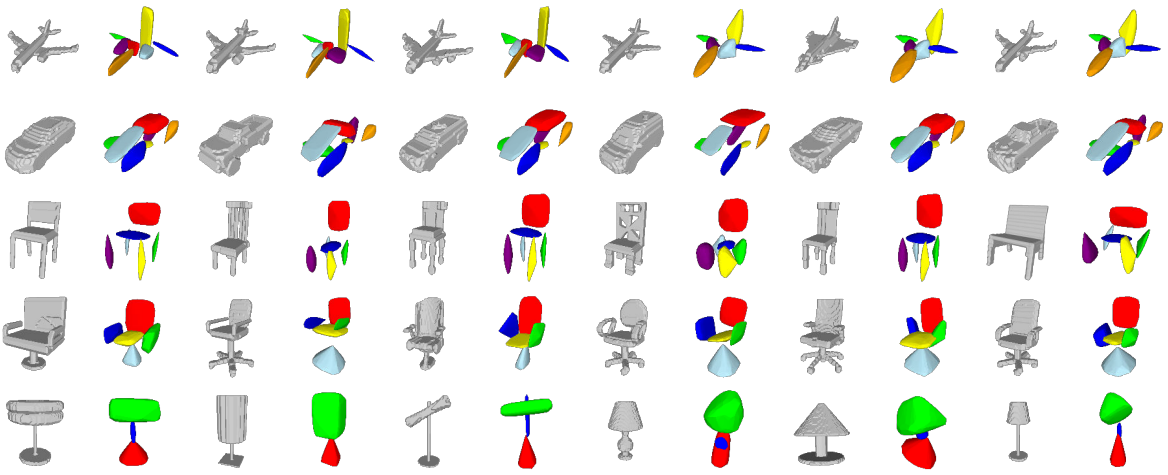


Figure 5: *Qualitative results of our unsupervised shape parsing on various shapes, where each pair is the original shape followed by the parsed shape with primitives colored according to their correspondence in the category. The number of primitives is set between* 2 *and* 8.

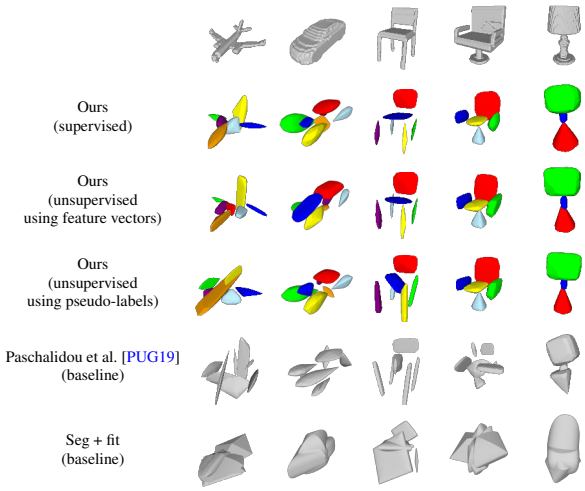


Figure 6: *Qualitative comparison between the supervised and unsupervised settings of our method and the baselines, with the corresponding original shape shown at the top of each column.*

4.3. Results

Figure 5 shows qualitative results of our method in the unsupervised setting, where we assign a color to each primitive to indicate the correspondence across the set. In the figure, we show the input meshes voxelized by marching cubes to give an idea of the input provided to the network. We see that our model successfully parses the shapes into meaningful components, approximating their general structure. Most of the approximations use the expected number of primitives. The correspondence is consistently predicted across various shapes within the same category.

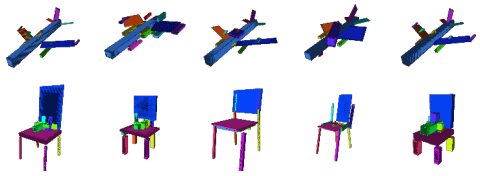


Figure 7: *Qualitative results from Yang and Chen [YC21], with each cuboid in the shape abstraction assigned a distinct color.*

Figure 6 presents a qualitative comparison between the supervised setting of our method, the two unsupervised variations of our method, one using the feature vectors extracted from the foundation model and the other using pseudo-labels for primitive fitting, and the two baselines, i.e., the method of Paschalidou et al. [PUG19] and "seg + fit". All these methods were trained using identical resources. When running our method, we found that the results showed no significant difference when using fewer than 200 shapes in the input set. Especially when both the training set and the initial number of primitives are small, our models generally achieve the expected number of instances/primitives and accurately approximate the structure of the shapes. In contrast, the results of the baselines provide less desirable approximations. On the other hand, the features extracted from the foundation model are not always fully informative, which implies that the results of our method are not perfect, particularly the results of our method using pseudo-labels, which can be expected for an unsupervised method.

Figure 7 shows qualitative results from the method of Yang and Chen [YC21]. Due to the architecture of their network, the method requires a large amount of training data to function effectively. As a result, we were only able to reproduce their results for two shape categories: airplanes and regular chairs. While the assembled cuboids capture the overall structure of the shapes, the method

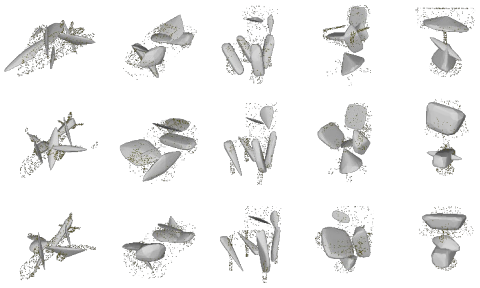


Figure 8: *Qualitative results of Paschalidou et al. [PUG19] when fixing the number of primitives to the expected number.*

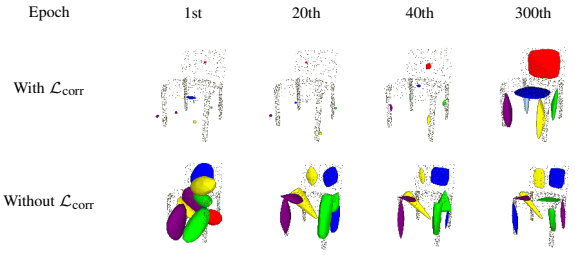
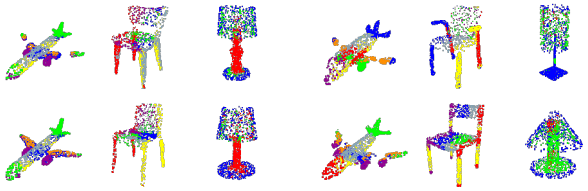


Figure 9: *Ablation study where we show results of intermediate iterations of the fitting. We use our full loss or a loss without $\mathcal{L}_{corr}$. With $\mathcal{L}_{corr}$, in the 20th iteration only 6 out of 8 primitives remain, while at the 40th iteration their fit starts to improve.*

tends to produce unnecessary extra cuboids, leading to visual artifacts in the reconstruction.

We also provide a quantitative comparison of the methods in Table 1, where the chamfer distance measures the reconstruction quality and PBRI measures the correctness of the correspondence (Section 4.2). Note that the scale of the chamfer distance is different from that reported by Paschalidou et al. [PUG19], due to factors such as shapes being normalized differently and a different number of superquadrics being fitted. We see that our method achieves the best or near-best performance among the unsupervised methods, and the results are close to the method that receives supervision for training. The method of Yang and Chen [YC21] yields strong reconstruction quality, as indicated by low chamfer distances. However, the unnecessary extra cuboids it generates reduce correspondence accuracy, resulting in lower PBRI scores.

Since our method requires the expected number of primitives k, we also evaluate the baseline of Paschalidou et al. [PUG19] by fixing the maximum number of primitives M to the same value k, which may be regarded as a fairer comparison to their method. We show example results in Figure 8, where the gray point clouds represent the input point clouds to which the primitives are being fit. We observe that the baseline fails to focus on any particular area, likely due to under-learning caused by the difficulty of discovering the optimal parameters with only a few superquadrics. In contrast, our approach succeeds by leveraging potential correspondences.



(a) *Typical features* **(b)** *Failure cases*

Figure 10: *Features extracted by the foundation model for selected example shapes.*

4.4. Ablation Study

We experiment with a variation of our method where we remove the correspondence loss $\mathcal{L}_{corr}$ from the formulation. Note that this is different from the baseline method as the baseline includes additional terms in the loss function that are not necessary in our method. In Figure 9, we show examples of how removing this loss impacts the quality of the results during different iterations of the optimization. We see that the training scheme described in Section 3.2 in addition to $\mathcal{L}_{corr}$ enables the method to move the primitives to suitable positions in earlier iterations, and then improve their fit in later iterations, which is not possible without simultaneously establishing correspondences.

4.5. Limitations

As discussed in Section 3.2, our method requires some user input to determine the best scheduling for dynamically adjusting the three weights of the loss function, so that we are able to obtain optimal results. Specifically, increasing the correspondence regularizer loss coefficient helps to achieve the desired number of primitives but also sharpens the predicted correspondence, causing it to focus on few features, which negatively impacts the approximation (similar to the results using pseudo-labels). Moreover, achieving good results also depends on the capabilities of the foundation model, which must provide features that effectively distinguish among different shape regions. Figure 10 shows typical examples where the foundation model provides features that distinguish different shape parts, contrasted to some failure examples where the features provided by the foundation model are less meaningful and may lead to errors in the fitting. Moreover, the use of k-means in the feature space creates clusters of points with similar features that are often spatially correlated. However, there is no guarantee that the points in the clusters are spatially connected.

5. Conclusion and Future Work

This paper presents a novel part-based 3D shape approximation method that jointly learns reconstruction and correspondence end-to-end. Our results and ablation study show that learning correspondences from start to finish reduces the risk of converging to suboptimal local minima. Our method achieves similar or better results than the baselines, requiring fewer initial primitives and faster convergence. We also introduce a novel correspondence loss, which can be used for unsupervised or supervised fitting, and the PBRI

	Reconstruction (chamfer distance↓)					Correspondence (PBRI↑)				
	Ours (S)	Ours (U)	[PUG19]	Seg + fit	[YC21]	Ours (S)	Ours (U)	[PUG19]	Seg + fit	[YC21]
Airplane	2.299	2.768	5.691	24.10	**1.103**	0.8000	**0.7137**	0.2792	0.6463	0.5685
Car	4.903	**5.373**	7.295	28.70	—	0.7425	**0.7692**	0.5140	0.6103	—
Chair (regular)	3.672	3.623	4.928	27.38	**1.289**	0.8798	**0.8460**	0.4491	0.6585	0.7321
Chair (office)	6.473	**4.724**	7.989	28.44	—	0.8239	**0.7890**	0.6309	0.6833	—
Lamp	3.579	**4.153**	5.730	61.20	—	0.7556	**0.7570**	0.6253	0.5374	—

Table 1: *Comparison of the mean chamfer distance (smaller is better) and spatial rand index (larger is better) for different methods across different categories, where the chamfer distance is multiplied by* 10^3. *"S" and "U" denote the supervised and unsupervised settings of our method, respectively. The best results per category and metric among the unsupervised methods are highlighted in bold.*

measure for evaluating the quality of primitive correspondence in the results. Future work includes extending the model with a self-adaptive tuning mechanism to further enhance its usability for automatic 3D shape parsing, and also explicitly enforcing the connectedness of points during clustering. It would also be interesting to explore the use of our method for matching parts of transformable objects [ZLH*25], e.g., for matching the parts of a sofa-bed in the sofa and bed transformation states.

Acknowledgments

We thank the anonymous reviewers for their valuable feedback. This work was partially supported by the Natural Sciences and Engineering Research Council of Canada.

References

[AMA23] ALANIZ S., MANCINI M., AKATA Z.: Iterative superquadric recomposition of 3D objects from multiple views. In *Proceedings of the IEEE International Conference on Computer Vision* (2023), pp. 17967–17977. 3

[Bie85] BIEDERMAN I.: Human image understanding: Recent research and a theory. *Computer Vision, Graphics, and Image Processing 32*, 1 (1985), 29–73. 1

[Bin71] BINFORD T. O.: Visual perception by computer. In *Proceedings of the IEEE Conference on Systems and Control* (1971). 1

[CCZZ24] CHEN Z., CHEN Q., ZHOU H., ZHANG H.: DAE-Net: Deforming auto-encoder for fine-grained shape co-segmentation. In *Proceedings of SIGGRAPH* (2024), pp. 82:1–82:11. 1

[CFG*15] CHANG A. X., FUNKHOUSER T., GUIBAS L., HANRAHAN P., HUANG Q., LI Z., SAVARESE S., SAVVA M., SONG S., SU H., XIAO J., YI L., YU F.: ShapeNet: An information-rich 3D model repository. *CoRR abs/1512.03012* (2015). 6

[CGF09] CHEN X., GOLOVINSKIY A., FUNKHOUSER T.: A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics 28*, 3 (2009), 73:1–73:12. 6

[CYF*19] CHEN Z., YIN K., FISHER M., CHAUDHURI S., ZHANG H.: BAE-NET: Branched autoencoder for shape co-segmentation. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 8489–8498. 3

[DMM24] DUTT N. S., MURALIKRISHNAN S., MITRA N. J.: Diffusion 3D features (Diff3F): Decorating untextured shapes with distilled semantic features. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2024), pp. 4494–4504. 3, 4

[GCS*20] GENOVA K., COLE F., SUD A., SARNA A., FUNKHOUSER T.: Local deep implicit functions for 3D shape. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2020), pp. 4856–4865. 1

[GCV*19] GENOVA K., COLE F., VLASIC D., SARNA A., FREEMAN W. T., FUNKHOUSER T.: Learning shape templates with structured implicit functions. In *Proceedings of the IEEE International Conference on Computer Vision* (2019), pp. 7153–7163. 1

[KAMC17] KALOGERAKIS E., AVERKIOU M., MAJI S., CHAUDHURI S.: 3D shape segmentation with projective convolutional networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 6630–6639. 1

[KLM*13] KIM V. G., LI W., MITRA N. J., CHAUDHURI S., DIVERDI S., FUNKHOUSER T.: Learning part-based templates from large collections of 3D shapes. *ACM Transactions on Graphics 32*, 4 (2013), 70:1–70:12. 2

[LASM24] LEI H., AKHTAR N., SHAH M., MIAN A.: Mesh convolution with continuous filters for 3-D surface parsing. *IEEE Transactions on Neural Networks and Learning Systems 35*, 10 (2024), 14863–14877. 1

[LHJ*22] LI Y., HE X., JIANG Y., LIU H., TAO Y., LIN H.: MeshFormer: High-resolution mesh segmentation with graph transformer. *Computer Graphics Forum 41*, 7 (2022), 37–49. 1

[LLY*23] LI Y., LIU S., YANG X., GUO J., GUO J., GUO Y.: Surface and edge detection for primitive fitting of point clouds. In *Proceedings of SIGGRAPH* (2023), pp. 44:1–44:10. 3

[LSC*21] LÊ E.-T., SUNG M., CEYLAN D., MECH R., BOUBEKEUR T., MITRA N. J.: CPFN: Cascaded primitive fitting networks for high-resolution point clouds. In *Proceedings of the IEEE International Conference on Computer Vision* (2021), pp. 7437–7446. 3

[LSD*19] LI L., SUNG M., DUBROVINA A., YI L., GUIBAS L.: Supervised fitting of geometric primitives to 3D point clouds. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 2647–2655. 3

[LWRC22] LIU W., WU Y., RUAN S., CHIRIKJIAN G. S.: Robust and accurate superquadric recovery: A probabilistic approach. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2022), pp. 2666–2675. 2

[LWRC23] LIU W., WU Y., RUAN S., CHIRIKJIAN G. S.: Marching-primitives: Shape abstraction from signed distance function. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2023), pp. 8771–8780. 1

[MZC*19] MO K., ZHU S., CHANG A. X., YI L., TRIPATHI S., GUIBAS L. J., SU H.: PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 909–918. 6

[NLX18] NIU C., LI J., XU K.: Im2Struct: Recovering 3D shape structure from a single RGB image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2018), pp. 4521–4529. 2

[ODM*24] OQUAB M., DARCET T., MOUTAKANNI T., VO H. V., SZAFRANIEC M., KHALIDOV V., FERNANDEZ P., HAZIZA D., MASSA F., EL-NOUBY A., ASSRAN M., BALLAS N., GALUBA W.,

HOWES R., HUANG P.-Y., LI S.-W., MISRA I., RABBAT M., SHARMA V., SYNNAEVE G., XU H., JÉGOU H., MAIRAL J., LABATUT P., JOULIN A., BOJANOWSKI P.: DINOv2: Learning robust visual features without supervision. *Transactions on Machine Learning Research 2024* (2024). 3

[Pen86] PENTLAND A.: Parts: Structured descriptions of shape. In *Proceedings of the AAAI Conference on Artificial Intelligence* (1986), pp. 695–701. 1, 2

[PUG19] PASCHALIDOU D., ULUSOY A. O., GEIGER A.: Superquadrics revisited: Learning 3D shape parsing beyond cuboids. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2019), pp. 10344–10353. 1, 2, 3, 4, 6, 7, 8, 9

[RBL*22] ROMBACH R., BLATTMANN A., LORENZ D., ESSER P., OMMER B.: High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2022), pp. 10674–10685. 3

[Roy23] ROY B.: Neural ShDF: Reviving an efficient and consistent mesh segmentation method. *CoRR abs/2306.11737* (2023). 1

[SSSS21] SHAKIBAJAHROMI B., SHAYESTEHMANESH S., SCHWARTZ D., SHOKOUFANDEH A.: HyNet: 3D segmentation using hybrid graph networks. In *Proceedings of the International Conference on 3D Vision* (2021), pp. 805–814. 1

[SZTL19] SUN C.-Y., ZOU Q.-F., TONG X., LIU Y.: Learning adaptive hierarchical cuboid abstractions of 3D shape collections. *ACM Transactions on Graphics 38*, 6 (2019), 241:1–241:13. 1

[TSG*17] TULSIANI S., SU H., GUIBAS L. J., EFROS A. A., MALIK J.: Learning shape abstractions by assembling volumetric primitives. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 1466–1474. 1, 2, 3

[WGS*18] WANG P., GAN Y., SHUI P., YU F., ZHANG Y., CHEN S., SUN Z.: 3D shape segmentation via shape fully convolutional networks. *Computers & Graphics 76* (2018), 182–192. 1

[YC21] YANG K., CHEN X.: Unsupervised learning for cuboid shape abstraction via joint segmentation from point clouds. *ACM Transactions on Graphics 40*, 4 (2021), 152:1–152:11. 2, 3, 6, 7, 8, 9

[YSGG17] YI L., SU H., GUO X., GUIBAS L. J.: SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2017), pp. 6584–6592. 1

[YYM*21] YAN S., YANG Z., MA C., HUANG H., VOUGA E., HUANG Q.: HPNet: Deep primitive segmentation using hybrid representations. In *Proceedings of the IEEE International Conference on Computer Vision* (2021), pp. 2733–2742. 3

[ZLH*25] ZHANG S.-K., LIU J.-H., HUANG J., CHI Z., TAM H., YANG Y.-L., ZHANG S.-H.: SceneExplorer: an interactive system for expanding, scheduling, and organizing transformable layouts. *IEEE Transactions on Visualization and Computer Graphics 31*, 9 (2025), 6008–6021. 9

[ZRA23] ZHANG L., RAO A., AGRAWALA M.: Adding conditional control to text-to-image diffusion models. In *Proceedings of the IEEE International Conference on Computer Vision* (2023), pp. 3836–3847. 3

[ZXC*20] ZHU C., XU K., CHAUDHURI S., YI L., GUIBAS L., ZHANG H.: AdaCoSeg: Adaptive shape co-segmentation with group consistency loss. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (2020), pp. 8540–8549. 3

[ZYY*17] ZOU C., YUMER E., YANG J., CEYLAN D., HOIEM D.: 3D-PRNN: Generating shape primitives with recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision* (2017), pp. 900–909. 2