# Iterative Superquadric Recomposition of 3D Objects from Multiple Views

Stephan Alaniz
University of Tübingen

Massimiliano Mancini
University of Trento

Zeynep Akata
University of Tübingen, MPI-IS

## Abstract

*Humans are good at recomposing novel objects, i.e. they can identify commonalities between unknown objects from general structure to finer detail, an ability difficult to replicate by machines. We propose a framework, ISCO, to recompose an object using 3D superquadrics as semantic parts directly from 2D views without training a model that uses 3D supervision. To achieve this, we optimize the superquadric parameters that compose a specific instance of the object, comparing its rendered 3D view and 2D image silhouette. Our ISCO framework iteratively adds new superquadrics wherever the reconstruction error is high, abstracting first coarse regions and then finer details of the target object. With this simple coarse-to-fine inductive bias, ISCO provides consistent superquadrics for related object parts, despite not having any semantic supervision. Since ISCO does not train any neural network, it is also inherently robust to out-of-distribution objects. Experiments show that, compared to recent single instance superquadrics reconstruction approaches, ISCO provides consistently more accurate 3D reconstructions, even from images in the wild. Code available at https://github.com/ExplainableML/ISCO.*

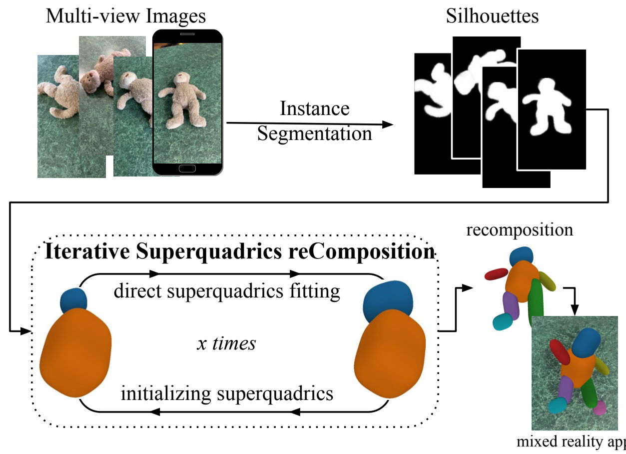


Figure 1: ISCO recomposes 3D objects from real-world images using superquadrics. It acts on 2D silhouettes and not 3D models learned via a neural network, placing new superquadrics iteratively into the scene. The final 3D abstraction and simple parts can then be used for mixed reality.

## 1. Introduction

Although the Jeff Koons sculpture "Balloon Dog" does not have a nose or teeth of a dog, we are able to recognize the dog when we look at the sculpture as a whole. This is because we can decompose the sculpture into semantically meaningful parts and recompose them in our minds to give it the name of a familiar object. Equipping machines with the same perceptual grouping capabilities would improve how they process and interact with the environment [46].

In this context, decomposing 3D objects by means of simple 3D shapes [45, 10] abstract away complex details of the object, providing a parsimonious representation useful in many applications, such as robotics [62] and virtual reality [35]. Recent works addressed this task by training a a neural network on a collection of 3D shapes, showing that semantically consistent part decompositions emerge, *i.e*. networks predict the same semantic parts (*e.g. back*, *legs*) across shapes (*e.g. chairs*, *sofa*) [60, 44, 43]. However, the data collection is costly in the real world, where the need for simple compositional representations is more evident. At the same time, instance-based approaches [29, 66] sidestep such need, but they require 3D inputs (*e.g*. point clouds) and do not extract semantically relatable parts.

In this work we take a different perspective, proposing Iterative Superquadric reComposition of Objects (ISCO), an instance-level self-supervised algorithm that, given multiple 2D views of an object, recomposes its 3D shape using superquadrics [45, 44], *without* access to 3D data. ISCO compares real and synthesized 2D views via a differentiable rendering pipeline [33], directly optimizing the primitive parameters. To extract more precise abstractions, we perform an iterative recomposition of the object i) initializing primitives in regions with high reconstruction error; ii) fitting them by prioritizing local accuracy over global coverage, representing finer details as the number of primitives increases. Results on ShapeNet [5] and ShapeNet-Part [74] show that ISCO achieves accurately reconstructs 3D shapes, identifying semantic parts much better than recent instance-based approaches[29, 66]. Acting at instance-level and on multiple views, ISCO can readily work on arbitrary objects and real-world images, as we show on Common Objects in 3D [51]. Fig. 1 shows a simplified version of our pipeline.

To summarize, our contributions are: i) we tackle the problem of 3D shape abstraction from multiple views, ii) we propose ISCO, a self-supervised algorithm that iteratively fits superquadrics to unexplained input regions, directly optimizing their parameters via a differentiable rendering pipeline; iii) we show that ISCO precisely reconstructs input shapes, is easily transferable to real-world applications and can better relate shape parts across different instances than previous instance-based approaches.

## 2. Related work

**3D Shape Decomposition.** Among different ways to represent 3D shapes, voxel grids [30], point clouds [48], triangular meshes [56], and signed distance functions [41] stand out. While they provide precise shape representations, their complexity results in limited interpretability. To address this issue works decomposed shapes into parts by using *e.g.* supervised objectives on part annotations [34, 48, 69, 6], zero/few-shot learning [31, 78, 55, 37], language [24, 13], or unsupervised objectives [60, 43, 14, 8, 7, 65, 71, 38].

In the unsupervised setting, most works decomposed 3D shapes using simple volumetric primitives since compact shape representations better aid shape analyses and parsing [60, 43, 52]. For instance, [60] predicts cuboids given an input 3D volume. [44] maps visual inputs (*e.g.* single-image, point clouds) to compositions of superquadrics, achieving a better trade-off between compactness and fidelity. Subsequent works explored more complex decompositions [42]), scenes [23], or atomic primitives (*e.g.* convexes [14], Gaussians [18, 17], planes [7], multi-tapered superquadrics [66], algebraic surfaces [73], deformed shapes [43]).

As in [44, 42] we still represent objects as composition of superquadrics, fitting a target shape with an unsupervised reconstruction objective. More closely related to us are early works fitting superquadrics to single instances given range data [25, 57], and especially more recent ones using point clouds as input [66, 29]. However, we do not need 3D inputs [66, 29], nor a training set as in [44]. Instead, we directly obtain abstractions from multiple views iteratively and via a differentiable rendering pipeline. Despite not using explicit 3D input, we show that ISCO better reconstructs the target shape than [66, 29] and provides semantic parsing results much closer to those of specific methods [8, 38].

**Neural Radiance Fields.** Since the seminal paper of Mildenhall et al. [33], NeRF has been the de-facto standard to synthesize novel views of a target scene. Practically, NeRF learns an implicit neural scene representation by mapping 3D coordinates and viewpoints to corresponding color and density values. This mapping is instance-specific and exploits a differentiable rendering pipeline to enforce consistency between the rendered and target views.

Subsequent works improved NeRF in multiple aspects, *e.g.* number of views [76, 36, 68], relative constraints [47, 32], surface reconstruction [39]synthesis quality [3, 19], scale [59, 61], and computational efficiency [50, 16, 20]. In this regard, recent works [58, 27, 75, 70] tried to replace the implicit scene representation of NeRF with explicit counterparts, using signed distance function [64], voxel grids [58], octrees [27, 75] or points [70]. While our method shares the same objective of fitting an explicit 3D representation to 2D views, we tackle the problem from a different perspective, *i.e.* recompose a target object with simple volumetric shapes, representing its parts. We achieve this by revisiting the original NeRF formulation, iteratively fitting superquadrics to the available 2D views.

Finally, our work is related to multi-view 3D reconstruction. This classical computer vision problem has various solutions, from Structure from Motion [53, 40], and SLAM [15] methods, to more recent multi-view stereo [4, 54] and learning-based [63, 67, 72] ones. While ISCO shares the same principles, our objective is different: providing an abstract composition of the target shape.

## 3. Superquadric Recomposition from 2D views

After formalizing our setting in Sec. 3.1, we describe our building blocks in Sec. 3.2. Finally, we describe our method that iteratively initializes and fits superquadrics in Sec. 3.3.

### 3.1. Problem definition

Our goal is to represent the volume of a static 3D object as a composition of simple volumetric shapes. When the 3D shape of an object is known, we can supervise this process by measuring how well the parts cover the ground-truth volume. While previous works [60, 44, 42, 43] assume the availability of such dense 3D data for supervision, in this work we relax this assumption and tackle the problem using images from multiple viewpoints around the object.

Formally, we have as input a set of images $\mathcal{I}$ depicting the target object from different known viewpoints. Our objective is to obtain a set of 3D parts $\mathcal{S}$ that best reconstructs the shape of the object as a whole. Note that, similarly to previous works [60, 44, 18, 14, 29, 66], our aim is to recompose the 3D shape of the target object, ignoring its particular texture or background. To achieve this, we supervise our model with silhouettes of the object extracted from the 2D views, *e.g.* using an out-of-the-box segmentation algorithm.

Without a ground-truth 3D representation, we cannot directly compare the compositions of our parts, *i.e.* superquadrics, to the target shape. However, we can use a ray-marching algorithm similar to [33] to render silhouettes

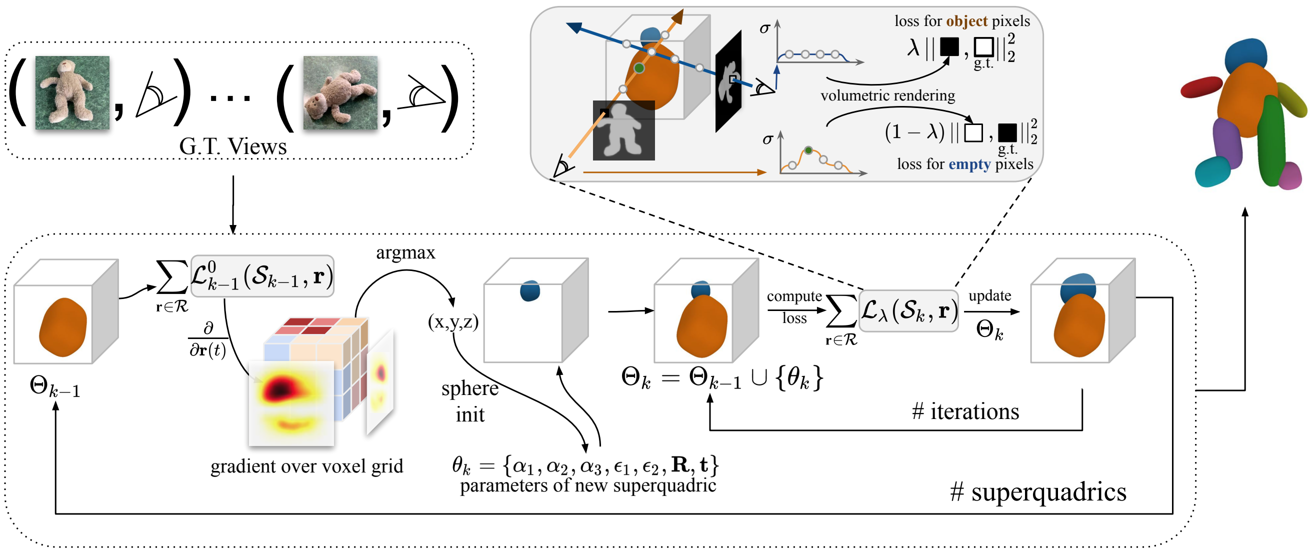


Figure 2: **Iterative Superquadric reComposition of Objects (ISCO)** recomposes 3D objects from 2D views. We iteratively initialize a new superquadric to the region where the estimated error is the highest (left) and optimize its shape parameters using differential rendering (right). The output is a 3D reconstruction of the object with superquadrics composing its parts.

of our superquadrics, in a fully differentiable manner, comparing the rendered silhouettes with the real ones in $\mathcal{I}$. Differently from [33], we do not train a neural network, but we directly update the superquadric parameters. In the following, we describe how we parameterize and render them.

### 3.2. Differentiable Superquadrics Rendering

**Superquadric representation.** The main building block of our algorithm is the simple volumetric 3D shape that we use to compose and represent the target object. We employ superquadrics [1] as primitives due to the variety of shapes they can represent with a simple parameterization that, as in [42, 29], has the implicit surface function $f$:

$$f(\mathbf{x};\theta) = \left(\left(\frac{x_1}{\alpha_1}\right)^{\frac{2}{\epsilon_2}} + \left(\frac{x_2}{\alpha_2}\right)^{\frac{2}{\epsilon_2}}\right)^{\frac{\epsilon_2}{\epsilon_1}} + \left(\frac{x_3}{\alpha_3}\right)^{\frac{2}{\epsilon_1}} \qquad (1)$$

where $\theta$ are the parameters describing a superquadric, with $\{\alpha_1, \alpha_2, \alpha_3\}$ being the scale along the coordinate axes, and $\{\epsilon_1, \epsilon_2\}$ the shape in its canonical form. A point $\mathbf{x}$ lies on the surface of the superquadric if $f(\mathbf{x};\theta) = 1$, inside if $f(\mathbf{x};\theta) < 1$, and outside if $f(\mathbf{x};\theta) > 1$.

To make the superquadrics more expressive, we define additional rigid transformation parameters:

$$\mathcal{T}(\mathbf{x};\theta) = \mathbf{R}(\theta)\mathbf{x} + \mathbf{t}(\theta) \qquad (2)$$

where $\mathbf{R}(\theta)$ is a 3D rotation matrix in Euler formulation and $\mathbf{t}(\theta)$ is a translation vector. Given a superquadric $p$, its set of parameters is thus $\theta_p = \{\alpha_1, \alpha_2, \alpha_3, \epsilon_1, \epsilon_2, \mathbf{R}, \mathbf{t}\}$. For simplicity, we will denote as $\hat{f}(\mathbf{x};\theta)$ the superquadric after applying the global transformations.

**Rendering superquadrics via ray-marching.** To render superquadrics, we define a density function $\sigma : \mathbb{R}^3 \to [0, 1]$

$$\sigma(\mathbf{x};\theta) = \texttt{sigmoid}\left(\gamma(1 - \hat{f}(\mathbf{x};\theta))\right) \qquad (3)$$

where $\gamma$ is a scalar determining the slope of the surface boundary. With a large $\gamma$, $\sigma(\mathbf{x};\theta)$ behaves like a step function that is 1 inside the superquadric and 0 outside, accurately reproducing its density. However, when fitting the superquadrics, we seek a smooth transition of $\sigma(\mathbf{x};\theta)$ to obtain well-behaved gradients, thus $\gamma$ is a hyperparameter.

With Eq. (3), we can now render superquadrics in 2D. Given a superquadric $s_k \in \mathcal{S}$, we adopt the volume rendering technique developed in [33] to obtain the density value along a given camera ray $\mathbf{r}(t) = \mathbf{o} + t\mathbf{d}$, with $\mathbf{o}$ being the origin and $\mathbf{d}$ its direction. We estimate the expected density of $s_k$ between near and far bounds $t_n$ and $t_f$ as

$$D(\mathbf{r}, s_k) = \int_{t_n}^{t_f} T(t)\sigma(\mathbf{r}(t), \theta_k)dt \qquad (4)$$

where $T(t) = \exp\left(-\int_{t_n}^{t} \sigma(\mathbf{r}(s), \theta_k)ds\right)$. Since we want to reconstruct 2D object silhouettes, we omit color values and directly use the density along the camera ray as grayscale pixel intensity. To numerically estimate this continuous integral, we use the same stratified sampling approach as in [33]. Note that, differently from [33], our $\sigma$ is not computed via a neural network, but directly computed from Eq. (3) using the parameters of the superquadrics.

### 3.3. Iterative Recomposition with Superquadrics

With Eq. (4), we can render our abstract representation given a viewpoint and use the discrepancy between rendered and real 2D views to fit the superquadrics to the target object. One straightforward approach is to randomly initialize a set of superquadrics and jointly optimize their parameters. However, this tends to find local optima, *e.g.* not covering all parts of the shape and/or the splitting a single part with multiple superquadrics (see Sec.4).

To address this issue, we propose an iterative pipeline, adding one superquadric at the time. We name our algorithm **I**terative **S**uperquadric re**C**omposition of **O**bject views (ISCO). Each iteration in ISCO has two steps: initialization of a new superquadric and its fitting. Fig 3 shows a simplified overview of the iterative fitting of superquadrics to a teddy bear, visualizing both the initialization step and the fitting process, with the final object representation being composition of the superquadrics in $\mathcal{S}_K$. In the following, we describe our objective function to fit a superquadric to ground-truth views, and how we use the same to initialize a new superquadric at the beginning of each iteration.

**Objective function.** Given a specific viewpoint, the rendered view of our superquadrics $\mathcal{S}_k$ at step $k$ should match the ground-truth one in $\mathcal{I}$. Thus, our objective function is

$$\mathcal{L}_k = \sum_{\mathbf{r}\in\mathcal{R}} ||D(\mathbf{r}, \mathcal{S}_k) - I(\mathbf{r})||_2^2. \tag{5}$$

where $\mathbf{r}$ is a camera ray in the set of all rays $\mathcal{R}$, $I(\mathbf{r})$ is the ground-truth (silhouette) value of the pixel for ray $\mathbf{r}$, and $D(\mathbf{r}, \mathcal{S}_k) = \min(\sum_{i=1}^{k} D(\mathbf{r}, s_i), 1)$. While $\mathcal{L}_k$ gives the same importance to pixels inside and outside the object, we found that it is beneficial to weight differently the two cases. We thus define our final objective at step $k$ as

$$\mathcal{L}_k^\lambda = \sum_{\mathbf{r}\in\mathcal{R}} \mathcal{L}_\lambda(\mathcal{S}_k, \mathbf{r}) = \sum_{\mathbf{r}\in\mathcal{R}} w_\lambda(\mathbf{r})||D(\mathbf{r}, \mathcal{S}_k) - I(\mathbf{r})||_2^2. \tag{6}$$

where $w_\lambda(\mathbf{r})$ is a weighting function with value $\lambda$ if the ray is outside the object (*i.e.* $I(\mathbf{r}) = 0$) and $1 - \lambda$ if it is inside (*i.e.* $I(\mathbf{r}) > 0$). With $\lambda$ we can achieve a better trade-off between covering large parts of the objects (*i.e.* $0 < \lambda < 0.5$) and/or focusing on local fitting (*i.e.* $\lambda > 0.5$). In the experiments, we follow the latter strategy, setting $\lambda = 0.6$, analyzing various $\lambda$ values in the supplementary.

**Superquadrics fitting via direct optimization.** The objective in Eq. (6) measures the discrepancy between rendered superquadrics and real views of the target object, allowing us to directly optimize the shapes in $\mathcal{S}_k$. This is possible thanks to i) the simple parameterization of superquadrics and ii) the differentiable rendering of Eq. (4). Given the set of superquadrics $\mathcal{S}_k$ with parameters $\Theta_k = \{\theta_1, \cdots, \theta_k\}$, we compute the gradient $\partial\mathcal{L}_k^\lambda/\partial\Theta_k$, and update the parameters via gradient descent:

$$\Theta_k \leftarrow \Theta_k + \texttt{optim}(\frac{\partial\mathcal{L}_k^\lambda}{\partial\Theta_k}) \tag{7}$$

where `optim` is an optimizer computing the specific update step from the gradient, *e.g.* Adam [21] in our experiments.

**Initialization by estimating missing parts.** The loss in Eq. (6) pushes the rendered and real views to match, but it does not guarantee that the superquadrics reconstruct all parts of the object after convergence. To encourage full coverage, we instantiate new superquadrics around object parts that have not been explained by any superquadric yet.

To estimate the position of missing parts, we use a dense voxel grid $G \in \mathbb{R}^{N\times N\times N}$, with resolution $N$, and we propagate rendering errors to the voxel grid. Specifically, we compute the gradient of $\mathcal{L}_{k-1}^{\lambda=0}$ w.r.t. the superquadrics density at every voxel $V_\mathbf{g} = \sigma(\mathbf{g}; \theta)$ with $\mathbf{g} \in G$. To do so, we render the superquadrics using ray marching as in Eq. (4), but instead of using the density value along the ray $\sigma(\mathbf{r}(t), \theta_k)$ we trilinearly interpolate the density from the voxel grid $V$ at the ray points $\mathbf{r}(t)$, the result of which we denote as $K_{\mathbf{r}(t)}$.We then propagate the gradients from $\mathcal{L}_{k-1}^{\lambda=0}$, where $\lambda = 0$ to consider only rays hitting the target object, to each $V_\mathbf{g}$ as

$$\frac{\partial\mathcal{L}_{k-1}^0}{\partial V_\mathbf{g}} = \sum_{\mathbf{r}\in\mathcal{R}} \sum_{\mathbf{r}(t)\in\mathbf{r}} \frac{\partial\mathcal{L}_{k-1}^0}{\partial K_{\mathbf{r}(t)}} \prod_{i=1}^{3} \max(0, 1 - \frac{|x_i^{(\mathbf{r}(t))} - x_i^{(\mathbf{g})}|}{l}) \tag{8}$$

where $[x_1^{(\mathbf{p})}, x_2^{(\mathbf{p})}, x_3^{(\mathbf{p})}]$ are the three-dimensional coordinates of point $\mathbf{p}$ and $l$ is the uniform distance between voxels in the grid. We provide a more detailed derivation in the supplementary.

Since the gradients over the voxel grid are an estimate of the regions with highest reconstruction errors, *i.e.* the higher the gradient and the higher the discrepancy between rendered and ground-truth views, we can use them to perform an informed initialization of a new superquadric. To initialize a new superquadric $s_k$, we first smooth $G$ with a Gaussian kernel, and then assign the translation parameters in $\theta_k$ to the coordinates of the maximum gradient value in the voxel grid, *i.e.* $\mathbf{t}(\theta)_k \leftarrow \arg\max_\mathbf{g} \partial\mathcal{L}_{k-1}^0/\partial V_\mathbf{g}$. We initialize the other parameters so that $s_k$ is a sphere, adding it to the set $\mathcal{S}_k$, *i.e.* $\mathcal{S}_k = \mathcal{S}_{k-1} \cup \{s_k\}$. Optimizing the superquadric starting from a region of high error ensures that we cover missing object parts, improving the quality of the abstraction as the number of superquadrics increases. To instantiate the first superquadric, we use the rendering error over an empty scene, *i.e.* $\mathcal{S}_0 = \emptyset$.

## 4. Experiments

In this section, we evaluate ISCO both quantitatively and qualitatively. We first describe the experimental setup, and then compare our method with related instance-based superquadrics fitting algorithms using point clouds [29, 66]. Furthermore, we ablate important aspects of our model, such as the number of views, superquadrics, and its iterative structure. Finally, we show results on semantic shape parsing and application on real world objects.

**Datasets.** To compare ISCO with previous works, we use ShapeNet [5] with the splits of [11], containing synthetic 3D models of 13 different classes, totaling more than 43k different instances. Having access to ground-truth shapes allows us to evaluate the accuracy of our reconstructions, even if we do not use 3D data in ISCO. To obtain 2D views, we sample 4, 8 or 16 different camera positions randomly from all angles around the object from which we render the respective silhouettes images. The camera poses are part of the input to ISCO and are required for the differentiable rendering step. We use the same procedure for evaluating semantic parsing performance, using the part annotations for ShapeNet provided in ShapeNet-part [74] and considering the same objects, parts, and evaluation protocol in [38].

Finally, to test real-world applicability of ISCO, we use the Common Objects in 3D (CO3D) [51] dataset. It contains multi-view images of around 19k objects from 50 MS-COCO [26] categories annotated by their corresponding camera viewpoints. CO3D illustrates a realistic pipeline, where we start from natural images, perform instance segmentation with PointRend [22] to obtain object masks, and apply ISCO as 3D abstraction method. For each scene we uniformly sample 16 random views along the camera trajectory from the video around the object.

**Implementation Details.** We tune the hyperparameters of our instance-based optimization process on the training set of ShapeNet, using the same set of hyperparameters for both the ShapeNet test set as well as ShapeNet-part, and CO3D. During the optimization procedure, we use 128x128 images, sample 500 rays per image, and perform 250 update steps before adding a new superquadric. Unless otherwise stated, we set $\lambda = 0.6$ and a maximum number of 10 superquadrics. Due to the iterative nature of ISCO, we obtain recompositions of the objects not only for 10 superquadrics, but any intermediate number as well with a single run. Our implementation is based on PyTorch3D [49].

**Baselines.** Since we perform instance-based 3D reconstruction, we compare our model with two recent state-of-the-art approaches, EMS [29] and NBP [66]. EMS [29] is a robust probabilistic algorithm for superquadrics recovery from point clouds, optimizing the superquadric parameters via the trust-region reflective algorithm [12] and avoiding local optima by searching for alternative parameters sets. NBP [66] extends [29] by fitting a mixture of tapered superquadrics [2] to the input point-cloud, alternating between points clustering and superquadrics optimization. Note that both algorithms use a point cloud as input, while we rely on multiple views. Since there are various algorithms for generating point clouds from multiple views, we ensure a fair comparison by using a *point cloud* sampled from the original mesh surface as input to EMS and NBP in the experiments on ShapeNet and ShapeNet-parts. Since in CO3D we do not have a ground-truth shape, we use [28] to fit a mesh from 16 views, sampling the relative point cloud. Additionally, we include NeuS [64], a neural implicit surface method that does not perform abstraction or object decomposition, but models the object implicitly with a neural network. NeuS serves as reference for non-abstract instance-based implicit shape representations. We also report results without the iterative procedure but initializing randomly all superquadrics (following the error of the empty scene $\mathcal{S}_0$) and optimizing them all at once (SCO).

| Input | Point Cloud | | 2D Views | | |
|---|---|---|---|---|---|
| Method | EMS [29] | NBP [66] | SCO | ISCO | NeuS [64] |
| airplane | 0.148 | 0.589 | 0.626 | **0.696** | 0.680 |
| bench | 0.066 | 0.443 | 0.489 | **0.499** | 0.630 |
| cabinet | 0.598 | 0.620 | 0.599 | **0.744** | 0.784 |
| car | 0.546 | 0.543 | 0.610 | **0.703** | 0.681 |
| chair | 0.148 | **0.595** | 0.526 | 0.549 | 0.706 |
| display | 0.358 | **0.635** | 0.629 | 0.631 | 0.738 |
| lamp | 0.174 | 0.520 | 0.404 | **0.531** | 0.593 |
| speaker | 0.599 | 0.643 | 0.576 | **0.756** | 0.776 |
| rifle | 0.261 | 0.556 | 0.584 | **0.692** | 0.736 |
| sofa | 0.319 | 0.606 | 0.601 | **0.694** | 0.780 |
| table | 0.083 | 0.568 | 0.487 | **0.576** | 0.610 |
| phone | 0.665 | 0.753 | 0.729 | **0.791** | 0.707 |
| vessel | 0.322 | 0.578 | 0.652 | **0.663** | 0.693 |
| mean | 0.330 | 0.588 | 0.578 | **0.656** | 0.701 |

Table 1: **Reconstruction on ShapeNet.** We report IoU of the object volume (higher is better). EMS and NBP use point cloud as input, ours and NeuS use 16 random views.

## 4.1. Shape Reconstruction on ShapeNet

In Tab. 1, we show the performance of ISCO, EMS, NBP, and SCO on 3D reconstruction on ShapeNet, where we use the same evaluation code of [42] to calculate the IoU of our superquadric reconstructions and the ground truth object meshes. ISCO consistently outperforms all others in average (*i.e.* 0.656 vs 0.588 of NBP), despite using 2D views and not point clouds as input. ISCO achieves the best results both in simpler objects (*e.g. speaker*, +0.11 on NBP) and more complex ones (*e.g.* +0.085 on NBP on *vessel*). Note also that NBP uses the more flexible tapered superquadrics but still achieves lower reconstruction accuracy than ISCO.

Interesting is also the gap with EMS and SCO that use the same superquadric parametrization and thus the same representation capacity. Our non-iterative pipeline (SCO), despite using the same input and optimization procedure, achieves less accurate reconstructions (*i.e.*-0.08 mIoU on

Figure 3: **Visualizing ISCO steps.** ISCO recomposes the object with a superquadric at a time, going from coarse to fine (top). The gradients of the reconstruction error on the voxel grid inform where to initialize the next one (bottom).


Figure 4: **ISCO with different number of views.** Objects are reconstructed with greater detail as we increase the views from 4 to 16, since they disambiguate more details.

| Views | 4 | | 8 | | 16 | |
|---|---|---|---|---|---|---|
| Method | SCO | ISCO | SCO | ISCO | SCO | ISCO |
| airplane | 0.520 | 0.578 | 0.594 | 0.657 | 0.626 | **0.696** |
| bench | 0.375 | 0.425 | 0.446 | 0.489 | 0.489 | **0.499** |
| cabinet | 0.565 | 0.730 | 0.578 | **0.746** | 0.599 | **0.744** |
| car | 0.531 | 0.637 | 0.598 | 0.680 | 0.610 | **0.703** |
| chair | 0.398 | 0.396 | 0.501 | 0.522 | 0.526 | **0.549** |
| display | 0.517 | 0.533 | 0.607 | 0.599 | 0.629 | **0.631** |
| lamp | 0.355 | 0.492 | 0.384 | 0.553 | 0.404 | **0.531** |
| speaker | 0.544 | 0.709 | 0.554 | 0.746 | 0.576 | **0.756** |
| rifle | 0.468 | 0.566 | 0.562 | 0.656 | 0.584 | **0.692** |
| sofa | 0.529 | 0.577 | 0.600 | 0.666 | 0.601 | **0.694** |
| table | 0.404 | 0.495 | 0.464 | 0.540 | 0.487 | **0.576** |
| phone | 0.629 | 0.739 | 0.702 | 0.770 | 0.729 | **0.791** |
| vessel | 0.569 | 0.611 | 0.645 | **0.661** | 0.652 | **0.663** |
| mean | 0.493 | 0.576 | 0.557 | 0.637 | 0.578 | **0.656** |

Table 2: **Analysis on the number of views.** We report IoU of the object volume on ShapeNet (higher is better). ISCO consistently outperforms its non-iterative version SCO.

average across all categories), showing the benefit of our iterative procedure. The large gap with EMS (*i.e.* 0.330 vs 0.656 on average) clearly shows how our initialization strategy and optimization procedure provide more precise feedback than existing probabilistic approaches. Overall, ISCO gets close to NeuS on average (0.656 vs. 0.701) and even surpasses it in some categories, such as airplane, car, and phone. Due to the limited amount of views, the inductive biases of superquadrics in ISCO can help modeling uncertain parts of the object which are not exposed by any view.

### 4.2. Analyzing and ablating ISCO on ShapeNet

In this section, we dissect our ISCO method to motivate its components and understand both its strengths and shortcomings. We analyze how the choice of the number of superquadrics and multi-view images affect its performance. Additionally, we study the impact of the iterative fitting process of ISCO to highlight its importance in inferring accurate superquadrics recompositions. We analyze the hyperparameter $\lambda$ in the supplementary.

**Number of views.** As we add more views from different angles of the scene, our 3D recomposition becomes more accurate as reported in Table 2. Even with as little as 4 views, our algorithm outperforms the average results of EMS (using point clouds) and SCO (at 16 views) from Tab. 1 (*i.e.* 0.576 vs 0.33 and 0.574 average IoU) and, with 8 views, the average results largely surpass also NBP (*i.e.* +0.05). Nevertheless, at a low number of views, it becomes more probable that parts of the object stay occluded and this may lead to poor reconstructions. In Fig. 4, we illustrate some examples for which we observe much lower IoU, especially at 4 views. Details and thinner object parts such as *chair* and *table legs* more often contain errors or are not covered by superquadrics due to potential ambiguities on the 2D views.

**Number of superquadrics.** ISCO adds superquadrics one

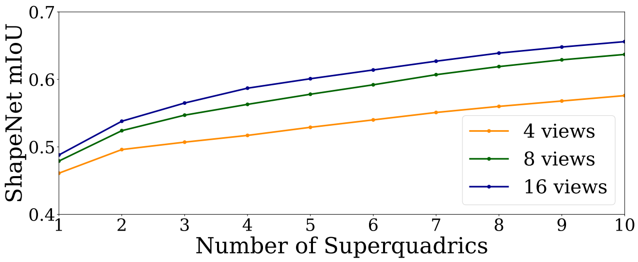


Figure 5: **Analysis on the number of superquadrics.** We report the mIoU on ShapeNet for increasing number of superquadrics during ISCO execution. Performance increase with the number of superquadrics.

by one to the scene making the reconstruction gradually more detailed and complex. For explainability purposes, we would like to describe the object with few shapes that match the key parts of the object rather than fitting every detail with increasingly smaller superquadrics. In this regard, ISCO is flexible to choose a subset of the maximum number of superquadrics to fit the target object, visualizing them after every iteration. Note that by focusing on local fitting with $\mathcal{L}_\lambda$ and initializing superquadrics at regions with high error, we obtain meaningful intermediate representations, even with few superquadrics.

In Fig. 3, we show how ISCO builds objects iteratively from their constituent parts. For each object, the top row depicts the superquadrics reconstruction throughout the algorithm where each image adds one superquadric. The row below shows a rendering of the error map we obtain by backpropagating gradients onto the volumetric voxel grid, guiding the placement of the next superquadric. We observe that coarse object parts are reconstructed first, with later superquadrics focusing on more fine-grained details, *e.g*. ISCO first represents the *seat* and *backrest* for the *chair*, the *body* and *wings* for the *airplane* and the *table top*.

From Fig. 5 we observe the gradual improvement in reconstruction fidelity (in IoU) as more superquadrics are added to the scene. With many superquadrics also comes diminishing returns at an additional cost of interpretability, which then contrasts with the goal of expose meaningful object parts, useful for downstream tasks. Therefore, we use a small number of superquadrics, *i.e*. 10.

**Iterative fitting.** In Fig. 6, we show some examples of different objects, where key object parts are correctly covered by a single superquadric in ISCO, and are made up of multiple ones in SCO. This harms the interpretability of individual shapes and their ability to semantically decompose an object. Moreover, details of the object are often not covered, i.e. legs of the *chair*. In this case, the targeted initializations in ISCO help reconstruct even fine details.

These results are corroborated by the overall performance drop in Tab. 1 and analyzed in Tab. 2 for 4 and 8 views. These qualitative and quantitative results demonstrate that, without the iterative procedure, superquadrics tend to compete to cover the whole object shape, without covering finer details. Our iterative procedure avoids this and allows each superquadric to focus on specific parts before introducing new elements in the scene.

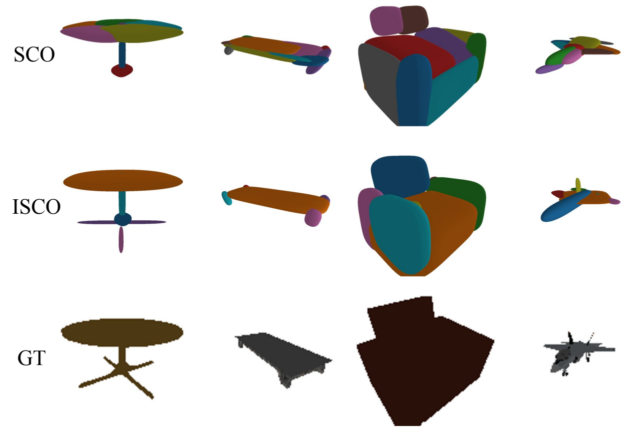


Figure 6: **The iterative procedure.** While ISCO precisely covers multiple object parts, in SCO superquadrics compete on the same regions to less interpretable recompositions.

| | MV | TF | C | plane 4 parts | chair 4 parts | table 2 parts |
|---|---|---|---|---|---|---|
| SQ [44] | | | ✓ | 48.9 | 65.6 | 77.7 |
| BAE [8] | | | ✓ | 61.1 | 65.5 | 87.0 |
| RIM [38] | | | ✓ | **67.8** | **81.5** | **91.2** |
| EMS [29] | | ✓ | | 33.4 | 45.6 | 46.9 |
| NBP [66] | | ✓ | | 33.8 | 53.4 | 44.3 |
| ISCO | ✓ | ✓ | | **67.7** | **76.6** | **81.9** |
| EMS [29] | | ✓ | ✓ | 26.0 | 38.1 | 32.3 |
| NBP [66] | | ✓ | ✓ | 23.1 | 38.7 | 32.9 |
| ISCO | ✓ | ✓ | ✓ | **33.9** | **55.5** | **70.7** |

Table 3: **Per-part mIoU (in %) on ShapeNet part [74].** Top: methods with point cloud input, using training set, and consistent part decomposition (C). Middle: training-set free (TF) and superquadric-based method with optimal semantic assignment. Bottom: the same method with fixed assignment. ISCO uses multiple views (MV) as input.

### 4.3. Semantic Parsing on ShapeNet-parts

We follow [38] and conduct an analysis of recovering object parts using ShapeNet part [74], measuring the performance as mIoU between the ground-truth shape part and the superquadrics assigned to it. In Tab. 3, we compare our model with EMS, NBP and the reference methods SQ [44], BAE [9] and RIM [38]. While SQ is a shape abstraction method using superquadrics, BAE and RIM are semantic parsing ones. In the table, all three methods use point clouds (*e.g.* as target in SQ, as input in BAE) and a training set of 3D shapes to achieve semantic parsing consistency.

Since ours, EMS and NBP perform instance-based abstraction and cannot directly assign semantics to parts, we considered two strategies for assigning semantic to superquadrics: *instance-wise*, where each superquadric is assigned to the closest object part, and *consistent*, where the order of the superquadrics determines the object part they are assigned to. In the latter case, the semantic identity of a given position is chosen as in [38] for unsupervised objectives.

Despite using weaker 3D information, our method outperforms EMS and NBP by a margin for all categories and both with and without consistent semantic assignments (*e.g.* with consistency, +8% mIoU on *plane*, +17% mIoU on *chair* and +38% mIoU on *table*). These results show the superiority of our model w.r.t. comparable superquadrics instance-based methods also in identifying semantic parts of the target shapes, even when consistency is enforced, showing the effectiveness of our iterative strategy reconstructing first coarse and then fine object regions.

Even though ISCO does not rely on learned per-class inductive biases as the competitors in the top part (*i.e.* SQ, BAE, RIM) it still obtains relatively good results when consistency is enforced (*e.g.*-7% w.r.t. SQ on *table*). The results without consistent assignments even surpass the training set-based competitors (*e.g.* +10% on BAE and SQ on *chair*) confirming the capability of ISCO in detecting the most relevant object parts, even if not always maintaining the same ordering. In fact, since ISCO is performed independently on each instance, there is no guarantee to obtain semantic consistency across instances of the same class, *e.g.* for the *plane* class the second and third superquadric could represent the left and right wing, respectively, but for another plane it might be the other way around. This causes the drop between the results with and without consistency.

### 4.4. Shape Recomposition of Real-world images

Using the Co3D, we can test ISCO and the baselines on recomposition of objects in the real-world. In Fig. 7, we present some qualitative examples of running ISCO, EMS, and NBP on scenes from CO3D that contain a *teddy bear*, a *toy plane* and a *toy truck*, a *laptop*, a *skateboard* and a *tv*. We select these classes since they contain more complex shapes than others in the dataset (*e.g. apple*, *ball*, *book*).

CO3D poses several additional challenges not present in the synthetic ShapeNet. To start with, object masks are obtained via instance segmentation with PointRend [22]. While the masks fit the object well on average, they contain noise and every set of images usually contains some artifacts, *e.g.* when the background color blends easily with

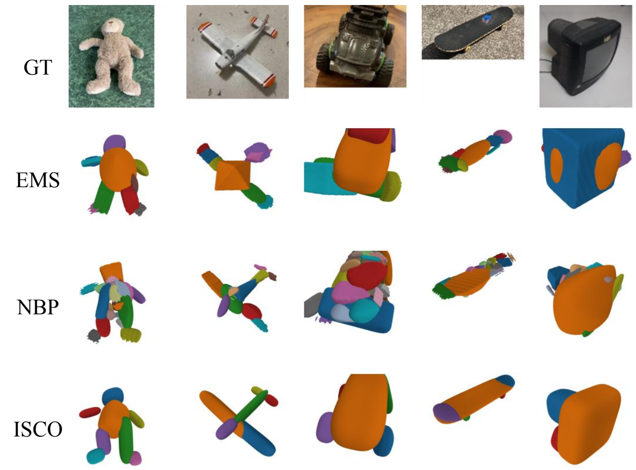


Figure 7: **Qualitative results on CO3D dataset.** We use 16 random views from each object, preprocessed to obtain their instance masks. Our method is the only one that can accurately rerecomposes the target object and its parts.

the object. These challenges extend to EMS and NBP, since reconstructed point clouds are more noisy from few views. Nevertheless, ISCO is generally robust to these type of noises, showing consistently better superquadrics recomposition than the competitors. Examples are the *tv*, where EMS covers the main body with two overlapped superquadrics, or *skateboard* where NBP uses several superquadrics to cover a small portion of the object.

We highlight that, while neural networks can be used to learn shape priors from training data [60, 44, 43], they to not generalize well to new shapes [77]. By performing instance-level training, ISCO is not affected by changes on classes or input distributions: for instance, it can decompose the *teddy bear* well into its parts, even if *e.g.* ShapeNet does not contain a class with similar anatomy to learn from.

## 5. Conclusion and Limitations

As a conclusion, we present Iterative Superquadric reComposition of Objects (ISCO), a self-supervised algorithm that recomposes 3D objects from multiple views using superquadrics. ISCO uses a differentiable rendering pipeline to fit parameters of superquadrics to the target scene by comparing rendered with real silhouettes. Superquadrics are added once at the time to the scene, initializing them over the regions with the highest estimated reconstruction error. This guarantees that superquadrics explain a dedicated object region well, before additional ones are optimized to further improve coverage. Results on ShapeNet and ShapeNet-parts show that ISCO is not only effective in reconstructing the target shape, but also in identifying its semantic parts, outperforming methods using point clouds as input. Lastly, we show the versatility of ISCO by applying it on in-the-wild objects from natural images of CO3D.

While the results show the effectiveness of ISCO, some acquisition conditions may impact the results. When views do not represent well the object, the reconstructions of ISCO may fail, *e.g.* the non-perfect wings of the *toy-plane* in CO3D (Fig. 7) due to lack of side views, or the ambiguous concavity of the rightmost *chair* on Fig. 4, that results in a unified seat/back. Extending ISCO to include shape priors and/or reconstruct texture and lighting conditions can mitigate this limitation. Another limitation is the computational cost. One update step on a 2080ti takes 13ms-22ms (for 4-16 views), obtaining a recomposition after 33s-55s. However, we did not optimize the runtime by *e.g.* choosing the number of superquadrics by early stopping when the initialization step finds a low reconstruction error. Note also that NBP has a comparable running time (*e.g.* 20-100s per instance but from point clouds) and that while EMS takes less than 1s per instance it still requires extracting a point cloud for real-world applications. Advances on neural rendering can also be applied to ISCO to further improve its runtime.

**Acknowledgements.** This work was supported by ERC (853489 - DEXIM), BMBF (Tübingen AI Center, FKZ: 01IS18039A), DFG (2064/1 – project number 390727645), and by the MUR PNRR project FAIR - Future AI Research (PE00000013) funded by the NextGenerationEU.

## References

[1] Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1(1):11–23, 1981. 3

[2] Alan H Barr. Global and local deformations of solid primitives. *ACM Siggraph Computer Graphics*, 18(3):21–30, 1984. 5

[3] Jonathan T Barron, Ben Mildenhall, Matthew Tancik, Peter Hedman, Ricardo Martin-Brualla, and Pratul P Srinivasan. Mip-nerf: A multiscale representation for anti-aliasing neural radiance fields. In *ICCV*, 2021. 2

[4] Neill DF Campbell, George Vogiatzis, Carlos Hernández, and Roberto Cipolla. Using multiple hypotheses to improve depth-maps for multi-view stereo. In *ECCV*, 2008. 2

[5] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. *CoRR*, abs/1512.03012, 2015. 1, 5

[6] Chao Chen, Yu-Shen Liu, and Zhizhong Han. Latent partition implicit with surface codes for 3d representation. In *ECCV*, 2022. 2

[7] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. In *CVPR*, 2020. 2

[8] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. Bae-net: Branched autoencoder for shape co-segmentation. In *ICCV*, 2019. 2, 7

[9] Zhiqin Chen, Kangxue Yin, Matthew Fisher, Siddhartha Chaudhuri, and Hao Zhang. BAE-NET: branched autoencoder for shape co-segmentation. In *ICCV*, 2019. 7

[10] Laurent Chevalier, Fabrice Jaillet, and Atilla Baskurt. Segmentation and superquadric modeling of 3d objects. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision (WSCG)*, 2003. 1

[11] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016. 5

[12] Thomas F Coleman and Yuying Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM Journal on optimization*, 6(2):418–445, 1996. 5

[13] Rodolfo Corona, Shizhan Zhu, Dan Klein, and Trevor Darrell. Voxel-informed language grounding. In *ACL (Volume 2: Short Papers)*, 2022. 2

[14] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. In *CVPR*, 2020. 2

[15] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial intelligence review*, 43:55–81, 2015. 2

[16] Stephan J Garbin, Marek Kowalski, Matthew Johnson, Jamie Shotton, and Julien Valentin. Fastnerf: High-fidelity neural rendering at 200fps. In *ICCV*, 2021. 2

[17] Kyle Genova, Forrester Cole, Avneesh Sud, Aaron Sarna, and Thomas Funkhouser. Local deep implicit functions for 3d shape. In *CVPR*, 2020. 2

[18] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *ICCV*, 2019. 2

[19] Yuan-Chen Guo, Di Kang, Linchao Bao, Yu He, and Song-Hai Zhang. Nerfren: Neural radiance fields with reflections. In *CVPR*, 2022. 2

[20] Tao Hu, Shu Liu, Yilun Chen, Tiancheng Shen, and Jiaya Jia. Efficientnerf efficient neural radiance fields. In *CVPR*, 2022. 2

[21] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 4

[22] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross B. Girshick. Pointrend: Image segmentation as rendering. In *CVPR*, 2020. 5, 8

[23] Florian Kluger, Hanno Ackermann, Eric Brachmann, Michael Ying Yang, and Bodo Rosenhahn. Cuboids revisited: Learning robust 3d shape fitting to single rgb images. In *CVPR*, 2021. 2

[24] Juil Koo, Ian Huang, Panos Achlioptas, Leonidas J Guibas, and Minhyuk Sung. Partglot: Learning shape part segmentation from language reference games. In *CVPR*, 2022. 2

[25] Ales Leonardis, Ales Jaklic, and Franc Solina. Superquadrics for segmenting and modeling range data. *IEEE TPAMI*, 1997. 2

[26] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence

Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 5

[27] Lingjie Liu, Jiatao Gu, Kyaw Zaw Lin, Tat-Seng Chua, and Christian Theobalt. Neural sparse voxel fields. *NeurIPS*, 2020. 2

[28] Shichen Liu, Tianye Li, Weikai Chen, and Hao Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *ICCV*, 2019. 5

[29] Weixiao Liu, Yuwei Wu, Sipu Ruan, and Gregory S Chirikjian. Robust and accurate superquadric recovery: a probabilistic approach. In *CVPR*, 2022. 1, 2, 3, 5, 7

[30] Zhijian Liu, Haotian Tang, Yujun Lin, and Song Han. Point-voxel cnn for efficient 3d deep learning. *NeurIPS*, 2019. 2

[31] Tiange Luo, Kaichun Mo, Zhiao Huang, Jiarui Xu, Siyu Hu, Liwei Wang, and Hao Su. Learning to group: A bottom-up framework for 3d part discovery in unseen categories. In *ICLR*, 2019. 2

[32] Ricardo Martin-Brualla, Noha Radwan, Mehdi SM Sajjadi, Jonathan T Barron, Alexey Dosovitskiy, and Daniel Duckworth. Nerf in the wild: Neural radiance fields for unconstrained photo collections. In *CVPR*, 2021. 2

[33] B Mildenhall, PP Srinivasan, M Tancik, JT Barron, R Ramamoorthi, and R Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020. 1, 2, 3

[34] Kaichun Mo, Shilin Zhu, Angel X Chang, Li Yi, Subarna Tripathi, Leonidas J Guibas, and Hao Su. Partnet: A large-scale benchmark for fine-grained and hierarchical part-level 3d object understanding. In *CVPR*, 2019. 2

[35] Konstantinos Moustakas, Georgios Nikolakis, Dimitrios Tzovaras, and Michael G Strintzis. A geometry education haptic vr application based on a new virtual hand representation. In *Virtual Reality*, pages 249–252. IEEE, 2005. 1

[36] Norman Müller, Andrea Simonelli, Lorenzo Porzi, Samuel Rota Bulò, Matthias Nießner, and Peter Kontschieder. Autorf: Learning 3d object radiance fields from single view observations. In *CVPR*, 2022. 2

[37] Muhammad Ferjad Naeem, Evin Pınar Örnek, Yongqin Xian, Luc Van Gool, and Federico Tombari. 3d compositional zero-shot learning with decompositional consensus. In *ECCV*, 2022. 2

[38] Chengjie Niu, Manyi Li, Kai Xu, and Hao Zhang. Rimnet: Recursive implicit fields for unsupervised learning of hierarchical shape structures. In *CVPR*, 2022. 2, 5, 7, 8

[39] Michael Oechsle, Songyou Peng, and Andreas Geiger. Unisurf: Unifying neural implicit surfaces and radiance fields for multi-view reconstruction. In *CVPR*, 2021. 2

[40] Onur Özyeşil, Vladislav Voroninski, Ronen Basri, and Amit Singer. A survey of structure from motion. *Acta Numerica*, 26:305–364, 2017. 2

[41] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *CVPR*, 2019. 2

[42] Despoina Paschalidou, Luc Van Gool, and Andreas Geiger. Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image. In *CVPR*, 2020. 2, 3, 5

[43] Despoina Paschalidou, Angelos Katharopoulos, Andreas Geiger, and Sanja Fidler. Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In *CVPR*, 2021. 1, 2, 8

[44] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *CVPR*, 2019. 1, 2, 7, 8

[45] Alex Pentland. Parts: Structured descriptions of shape. In *AAAI*, 1986. 1

[46] Alex P Pentland. Perceptual organization and the representation of natural form. In *Readings in Computer Vision*, pages 680–699. Elsevier, 1987. 1

[47] Albert Pumarola, Enric Corona, Gerard Pons-Moll, and Francesc Moreno-Noguer. D-nerf: Neural radiance fields for dynamic scenes. In *CVPR*, 2021. 2

[48] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *CVPR*, 2017. 2

[49] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 5

[50] Christian Reiser, Songyou Peng, Yiyi Liao, and Andreas Geiger. Kilonerf: Speeding up neural radiance fields with thousands of tiny mlps. In *ICCV*, 2021. 2

[51] Jeremy Reizenstein, Roman Shapovalov, Philipp Henzler, Luca Sbordone, Patrick Labatut, and David Novotny. Common objects in 3d: Large-scale learning and evaluation of real-life 3d category reconstruction. In *ICCV*, 2021. 2, 5

[52] Daxuan Ren, Jianmin Zheng, Jianfei Cai, Jiatong Li, and Junzhe Zhang. Extrudenet: Unsupervised inverse sketch-and-extrude for shape parsing. In *ECCV*, 2022. 2

[53] Johannes L Schonberger and Jan-Michael Frahm. Structure-from-motion revisited. In *CVPR*, 2016. 2

[54] Johannes L Schönberger, Enliang Zheng, Jan-Michael Frahm, and Marc Pollefeys. Pixelwise view selection for unstructured multi-view stereo. In *ECCV*, 2016. 2

[55] Gopal Sharma, Bidya Dash, Aruni RoyChowdhury, Matheus Gadelha, Marios Loizou, LiangLiang Cao, Rui Wang, Learned-Miller Erik, Subhransu Maji, and Evangelos Kalogerakis. Prifit: Learning to fit primitives improves few shot point cloud segmentation. In *Eurographics Symposium on Geometry Processing*, 2022. 2

[56] Jonathan Richard Shewchuk. Delaunay refinement algorithms for triangular mesh generation. *Computational geometry*, 22(1-3):21–74, 2002. 2

[57] Franc Solina and Ruzena Bajcsy. Recovery of parametric models from range images: The case for superquadrics with global deformations. *IEEE TPAMI*, 1990. 2

[58] Cheng Sun, Min Sun, and Hwann-Tzong Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *CVPR*, 2022. 2

[59] Matthew Tancik, Vincent Casser, Xinchen Yan, Sabeek Pradhan, Ben Mildenhall, Pratul P Srinivasan, Jonathan T Barron, and Henrik Kretzschmar. Block-nerf: Scalable large scene neural view synthesis. In *CVPR*, 2022. 2

[60] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *CVPR*, 2017. 1, 2, 8

[61] Haithem Turki, Deva Ramanan, and Mahadev Satyanarayanan. Mega-nerf: Scalable construction of large-scale nerfs for virtual fly-throughs. In *CVPR*, 2022. 2

[62] Giulia Vezzani, Ugo Pattacini, and Lorenzo Natale. A grasping approach based on superquadric models. In *ICRA*, 2017. 1

[63] Dan Wang, Xinrui Cui, Xun Chen, Zhengxia Zou, Tianyang Shi, Septimiu Salcudean, Z Jane Wang, and Rabab Ward. Multi-view 3d reconstruction with transformers. In *ICCV*, 2021. 2

[64] Peng Wang, Lingjie Liu, Yuan Liu, Christian Theobalt, Taku Komura, and Wenping Wang. Neus: Learning neural implicit surfaces by volume rendering for multi-view reconstruction. *NeurIPS*, 2021. 2, 5

[65] Xiaogang Wang, Xun Sun, Xinyu Cao, Kai Xu, and Bin Zhou. Learning fine-grained segmentation of 3d shapes without part labels. In *CVPR*, 2021. 2

[66] Yuwei Wu, Weixiao Liu, Sipu Ruan, and Gregory S Chirikjian. Primitive-based shape abstraction via nonparametric bayesian inference. In *ECCV*, 2022. 1, 2, 5, 7

[67] Haozhe Xie, Hongxun Yao, Xiaoshuai Sun, Shangchen Zhou, and Shengping Zhang. Pix2vox: Context-aware 3d reconstruction from single and multi-view images. In *ICCV*, 2019. 2

[68] Dejia Xu, Yifan Jiang, Peihao Wang, Zhiwen Fan, Humphrey Shi, and Zhangyang Wang. Sinnerf: Training neural radiance fields on complex scenes from a single image. In *ECCV*, 2022. 2

[69] Mutian Xu, Junhao Zhang, Zhipeng Zhou, Mingye Xu, Xiaojuan Qi, and Yu Qiao. Learning geometry-disentangled representation for complementary understanding of 3d object point cloud. In *AAAI*, 2021. 2

[70] Qiangeng Xu, Zexiang Xu, Julien Philip, Sai Bi, Zhixin Shu, Kalyan Sunkavalli, and Ulrich Neumann. Point-nerf: Point-based neural radiance fields. In *CVPR*, 2022. 2

[71] Chun-Han Yao, Wei-Chih Hung, Varun Jampani, and Ming-Hsuan Yang. Discovering 3d parts from image collections. In *ICCV*, 2021. 2

[72] Lior Yariv, Yoni Kasten, Dror Moran, Meirav Galun, Matan Atzmon, Basri Ronen, and Yaron Lipman. Multiview neural surface reconstruction by disentangling geometry and appearance. *NeurIPS*, 2020. 2

[73] Mohsen Yavartanoo, Jaeyoung Chung, Reyhaneh Neshatavar, and Kyoung Mu Lee. 3dias: 3d shape reconstruction with implicit algebraic surfaces. In *ICCV*, 2021. 2

[74] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas Guibas. A scalable active framework for region annotation in 3d shape collections. *SIGGRAPH Asia*, 2016. 1, 5, 7

[75] Alex Yu, Ruilong Li, Matthew Tancik, Hao Li, Ren Ng, and Angjoo Kanazawa. Plenoctrees for real-time rendering of neural radiance fields. In *ICCV*, 2021. 2

[76] Alex Yu, Vickie Ye, Matthew Tancik, and Angjoo Kanazawa. pixelnerf: Neural radiance fields from one or few images. In *CVPR*, 2021. 2

[77] Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Josh Tenenbaum, Bill Freeman, and Jiajun Wu. Learning to reconstruct shapes from unseen classes. *NeurIPS*, 2018. 8

[78] Yongheng Zhao, Tolga Birdal, Haowen Deng, and Federico Tombari. 3d point capsule networks. In *CVPR*, 2019. 2