DPA-Net: Structured 3D Abstraction from Sparse Views via Differentiable Primitive Assembly

Fenggen Yu[1,2⋆], Yiming Qian[1], Xu Zhang[1], Francisca Gil-Ureta[1], Brian Jackson[1], Eric Bennett[1], and Hao Zhang[1,2]

[1] Amazon
[2] Simon Fraser University

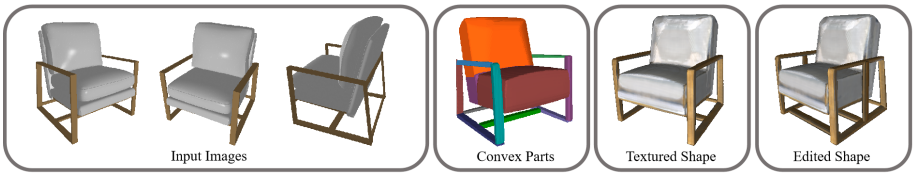

Fig. 1: Our method takes as few as three RGB images from disparate views and abstracts a textured 3D shape formed by a union of convex parts that well reflect the shape semantics. With a differentiable primitive assembly subject to only image-space losses, our network is trained without 3D supervision. With the meaningful parts abstracted, the resulting shape can be edited.

Abstract. We present a *differentiable rendering* framework to learn *structured 3D abstractions* in the form of *primitive assemblies* from sparse RGB images capturing a 3D object. By leveraging differentiable volume rendering, our method does not require 3D supervision. Architecturally, our network follows the general pipeline of an image-conditioned neural radiance field (NeRF) exemplified by pixelNeRF for color prediction. As our core contribution, we introduce *differential primitive assembly* (DPA) into NeRF to output a 3D occupancy field in place of density prediction, where the predicted occupancies serve as opacity values for volume rendering. Our network, coined DPA-Net, produces a union of convexes, each as an intersection of convex quadric primitives, to approximate the target 3D object, subject to an abstraction loss and a masking loss, both defined in the image space upon volume rendering. With test-time adaptation and additional sampling and loss designs aimed at improving the accuracy and compactness of the obtained assemblies, our method demonstrates superior performance over state-of-the-art alternatives for 3D primitive abstraction from sparse views.

1 Introduction

3D reasoning, e.g., abstraction or reconstruction, from single or multi-view images is one of the most fundamental problems in computer vision. With the recent emergence of neural fields [48], especially neural radiance fields (NeRF) [26, 30] and 3D Gaussian splatting [20], rapid advances have been made in 3D reconstruction quality, speed, and the ability to take on sparse [25, 46, 53, 56] rather than dense input views as in earlier

⋆ Work carried out during internship at Amazon.

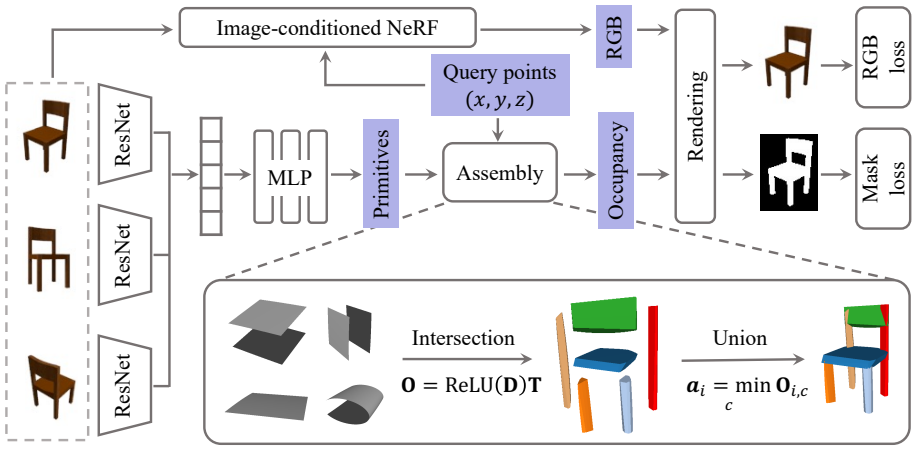

Fig. 2: Overview of DPA-Net. Given a sparse set of input RGB images whose viewpoints can be significantly different, our network is trained to predict a 3D primitive assembly, i.e., a 3D abstraction, via differentiable volume rendering without 3D supervision. The high-level network architecture resembles that of an image-conditioned NeRF such as pixelNeRF [53] for color prediction from multi-scale image features. What is new is that the density estimation in NeRF is replaced by our novel differentiable primitive assembly (DPA). DPA takes as input multi-view image features from ResNet that are fused into a shape feature via weighted pooling. The shape feature is further passed into an MLP (the primitive decoder) to predict the parameters of a set of convex quadric primitives. 3D query points and the primitives are assembled by two CSG-based assembly layers (intersection and then union) to predict point occupancies, which serve as opacity values for both volume rendering and for predicting an image mask. An RGB loss and a masking loss are calculated against the input images and object masks. Note that we assume that the camera poses and object masks are either provided or estimated prior to our 3D abstraction.

works. However, by design, NeRF and most of its variants target novel view synthesis with a focus on optimizing their primitives to improve *rendering* performances, rather than serving downstream tasks involving shape *modeling* or *manipulation*.

Higher-level primitives than points and Gaussian splats are more suited to model shape structures [27], facilitating structure-level analysis and synthesis tasks such as semantic labeling, interactive shape editing, assembly-based modeling, and visual program induction [4]. Recently, several methods have been proposed for CAD modeling by learning primitive assemblies such as constructive solid geometry (CSG) trees [8, 18, 54, 55], sketch-n-extrude models [22], or shape programs [11, 17]. However, these neural models all take 3D inputs such as voxels and point clouds.

In this paper, we introduce a learning framework which outputs a *3D abstraction*, in the form of a *primitive assembly*, from a small number of RGB images capturing a 3D object. Importantly, the sparse input images are not only few in number, i.e., as few as *three*, but also *disparate* [46], meaning that the viewpoints may be significantly different; see Fig. 1. In addition to tackling this challenge, our method also does not require 3D supervision for the multi-view 3D abstraction. It resorts to *differentiable rendering* of the assembled 3D primitives to define image-space abstraction losses.

Architecturally, our network follows the general pipeline of an image-conditioned NeRF (see Fig. 2), such as pixelNeRF [53]. In addition to 3D positions and camera ray information, we incorporate multi-scale image features from a ResNet [12] encoder applied to a training set for differentiable rendering, enabling generalization across scenes.

Our new and core contribution is the introduction of *differential primitive assembly* (DPA) into the NeRF framework to output a *3D occupancy field* in place of density prediction, so that the predicted occupancies would serve as opacity values for volume rendering. Specifically, our network, coined DPA-Net, produces a union of convexes, each as an intersection of convex quadric primitives, to approximate the target 3D object. The entire process requires no 3D models for supervision; it takes fused multi-view image features as well as NeRF inputs to predict parameters for the CSG primitive assembly and the occupancy field, subject to an abstraction loss (named as RGB loss in Fig. 2) and a masking loss, both defined in the image space upon volume rendering.

Furthermore, the abstraction of *clean* and *structurally* accurate primitive assemblies from sparse views offers several special challenges. To address them, we incorporate three enhancements: a) a primitive *dropout* scheme to improve compactness of the assembly; b) an adaptive point sampling to assist in the recovery of thin structures; and c) an additional penalty to discourage primitive overlap.

To assess our method, we train it on multi-view projections of 3D models from ShapeNet [3] as well as real images from the DTU dataset [15] for sparse-view abstraction. We conduct qualitative and quantitative evaluations, ablation studies, as well as comparisons to state-of-the-art alternatives for 3D primitive abstraction from sparse-view images, demonstrating superior performance of DPA-Net. The applications further demonstrate that users can easily edit our 3D abstraction results in popular CAD softwares, e.g., MeshLab [7] and OpenSCAD [42]. The editable abstractions can serve as structural prompts and benefit other 3D generation tasks [5, 40].

2 Related Works

The literature on multi-view 3D reasoning and representation learning of 3D structures and abstractions is vast. In this section, we focus on prior works that are closely related to our DPA approach, especially those designed to handle sparse input views.

2.1 Multi-view 3D reconstruction

NeRF [26], most of its variants, and other multi-view neural 3D reconstruction methods have been designed for dense views [48], but there are exceptions. SparseNeUS [25] can take on input views that are few in number, but *close-by*, whereas VolRecon [39] improves the reconstruction quality with additional ground-truth (GT) depth maps as supervision. Fewer methods have been designed to handle sparse *and disparate* views. In particular, FvOR [52] jointly solves for camera poses and 3D geometry while requiring GT 3D shapes to supervise. FORGE [16] improves FvOR by leveraging cross-view correlation, while using 2D images as supervision. Most recently, DiViNet [46] learns Gaussian surface priors, called neural templates, during training. At test times, it applies these templates as anchors to help stitch the surfaces over sparse regions. While FORGE

and DiViNet both employ differentiable volume rendering as in our approach, their results are neither compact, in terms of primitive/Gaussian counts, nor well-structured to reflect shape semantics, as in the primitive assemblies obtained by our work.

As described above, our network architecture resembles that of pixelNeRF [53], which is an early and representative method for generalizable (i.e., not an overfit model) NeRF prediction from very few input views. Since then, there have been more recent works for sparse-view NeRFs, *e.g.*, IBRNet [47], RegNeRF [31], depth-supervised NeRF [10], FreeNeRF [51], SPARF [44], including those designed to handle single-view images, *e.g.*, Pix2NeRF [2] and Zero-1-to-3 [23]. Comparing abstraction results from DPA-Net and those of pixelNeRF, we observe that their overall accuracies are comparable, but when the textures are removed, the mesh surfaces obtained from pixelNeRF results are quite noisy. It is important to note that our work is *not* intended to outperform all state-of-the-art methods on sparse-view 3D reconstruction in terms of reconstruction quality. Foremost, we target neural 3D *primitive assembly/abstraction* while aiming to achieve a high reconstruction quality. For reference, we compare DPA-Net to both pixelNeRF and DiViNeT in Section 5.

2.2 Learning 3D structures and abstractions

The dominant majority of prior works on learning 3D shape structures [4] take on 3D inputs, *e.g.*, voxel grids, polygonal meshes, point clouds. This is also true for 3D abstractions designed for a variety of primitives such as cuboids [45], super quadrics [24, 37], and convexes [6, 9]. Among them, EMS (for Expectation, Maximization, and Switching) is a probabilistic method to recover superquadrics from point clouds, which can be obtained from pure image inputs via multi-view stereo (MVS). Hence, it is chosen as a baseline to evaluation our method in Section 5.

Most relevant to our approach are the series of works on unsupervised learning of CSG assemblies, including UCSG [18], CAPRI-Net [55], CSG-Stump [8], and most recently, D^2CSG [54], all of which optimize a CSG tree from 3D inputs. Our quadric-based primitive assembly is built on CAPRI-Net [55], with our core contribution being to integrate it with differential volume rendering for DPA from few and disparate views.

Earlier works on structured CAD modeling or abstraction from images [13,14,49] or sketches [50] all require 3D models to either form a repository for shape/part retrieval or serve as training data for supervised learning. Im2Struct [33] recovers 3D shape structures, in the form of a cuboid assembly, from a single RGB image. Binary object masks were also used as an auxiliary signal to assist in the structure recovery from input images, where the latter was trained with ground-truth 3D cuboid assemblies. Also requires 3D models for training, StructureNet [28] jointly embeds 3D cuboid representations and images into a common latent space to allow 3D cuboid abstractions from images. With 3D supervision, BSP-Net [6] is able to predict a plane assembly from single-view RGB images. Furthermore, RIM-Net [34] and [35] learn hierarchical part decompositions under 3D reconstruction supervision.

To our knowledge, only three very recent works [1,29,43] have attempted 3D primitive abstraction from RGB images without 3D supervision. PartNeRF [43] learns a 3D primitive abstraction using ellipsoids, targeting 3D shape generation rather than reconstruction. Their use of ellipsoidal primitives also limits the method's ability to learn

a meaningful part decomposition. For example, the top of a table is always split into multiple parts. ISCO [1] optimizes parameters for 3D superquadrics to recompose an object directly from 2D views, where the rendering loss is defined with respect to 2D image silhouettes. On few input views, *e.g.*, four, their abstraction results were admittedly "poor"; see Section 4.2 of their paper. Since there is no published code by ISCO, we mainly compare our method with Differentiable Blocks World (DBW) [29], which can produce textured superquadric meshes. Overall, their results are compact, *i.e.*, with few primitives, but the abstraction errors are relatively high.

3 Approach

Given a small number of RGB images depicting a single object from disparate camera views, our goal is to learn a 3D shape abstraction assembled from a set of simple geometric primitives (quadrics in this paper). We assume that the camera parameters and object mask for each view are known or pre-estimated. The cameras exhibit wide baselines and the number of images is typically less than six.

Our method does not require any 3D supervision such as GT primitive decompositions. The differentiable primitive assembly facilitates volume rendering and is trained with image-space losses only, as shown in Fig. 2.

3.1 Feature extraction and aggregation

For each input image, we employ ResNet34 [12] as the feature encoder and alter its final linear layer to output a 256-dimensional feature vector. We then utilize the weighted pooling proposed in [47] to obtain an aggregated feature $\mathbf{f} \in \mathbb{R}^{256}$ from multi-view images. In practice, the weights are computed with a shared single-layer multi-layer-perceptron (MLP) followed by a softmax function.

3.2 Primitive assembly

Inspired by the state-of-the-art (SOTA) 3D primitive representation CAPRI-Net [55], we model a 3D shape as the assembly of a set of *quadric* primitives using differentiable intersection and union operations. This step takes in the fused image feature $\mathbf{f}$ and a set of query points evenly sampled from camera rays as in NeRF [26], and predicts primitive parameters and an occupancy field of the shape—whether a query point is inside or outside the shape. It is noteworthy that our intersection and union operations are adopted from [6, 55]; we briefly discuss them here for completeness.

Primitive parameterization: We define a quadric primitive as: $|a|x^2 + |b|y^2 + |c|z^2 + dx + ey + fz + g = 0$, where (x, y, z) is a query point and (a, b, c, d, e, f, g) are the primitive parameters. We enforce the first three parameters to be non-negative to include convex quadric primitives only. Denote the number of primitives as P, we obtain a matrix $\mathbf{P} \in \mathbb{R}^{P\times 7}$ by stacking all primitive parameters. Then a 2-layer MLP is applied to infer $\mathbf{P}$ from the fused feature $\mathbf{f}$. We set $P = 4,096$ in this paper and please refer to the supplementary for an ablation study.

Define matrix $\mathbf{Q}$ whose ith row $(x_i^2, y_i^2, z_i^2, x_i, y_i, z_i, 1)$ is computed from the ith query point (x_i, y_i, z_i). With N query points, we estimate an $N \times P$ matrix $\mathbf{D} = \mathbf{Q}\mathbf{P}^\top$, where $\mathbf{D}_{i,p}$ denotes the approximate signed distance between the ith point and the pth primitive. We have $\mathbf{D}_{i,p} < 0$ if point i is *inside* the primitive and $\mathbf{D}_{i,p} > 0$ if *outside*.

Primitive intersection: We adaptively group the P quadric primitives into $C = 256$ convexes. Our goal is to check whether a query point is inside any convex. To this end, we utilize a *learnable* selection matrix $\mathbf{T} \in \mathbb{R}^{P \times C}$. Ideally, $\mathbf{T}$ is binary, but may take floating-point values in $[0, 1]$ depending on the training phase; see Sec. 3.4. Each column in $\mathbf{T}$ selects a subset of primitives whose intersection forms one of the C convexes. In practice, we calculate an $N \times C$ matrix $\mathbf{O} = \text{ReLU}(\mathbf{D})\mathbf{T}$, with the ReLU function. We set $\mathbf{O}_{i,c} = 0$ if the ith point is inside the cth convex, and $\mathbf{O}_{i,c} > 0$ if outside.

Union of convexes: We further group the convexes to form the final 3D shape, checking whether a query point is inside this shape (*i.e.*, the occupancies). We attain this with a min-pooling among all convexes for the ith point, analogous to the union in CSG:

$$\mathbf{a}_i^* = \min_{1 \le c \le C} \mathbf{O}_{i,c} \quad \begin{cases} = 0 & \text{inside,} \\ > 0 & \text{outside,} \end{cases} \tag{1}$$

where $\mathbf{a}^* \in \mathbb{R}^N$ denotes the occupancy field. In practice, a soft approximation is calculated for better gradient flow:

$$\mathbf{a}_i^+ = \mathscr{L}\left(\sum_{c=1}^{C} \mathbf{w}_c \mathscr{L}\left(1 - \mathbf{O}_{i,c}\right)\right) \quad \begin{cases} = 1 & \approx \text{inside,} \\ < 1 & \approx \text{outside,} \end{cases} \tag{2}$$

where $\mathbf{w} \in \mathbb{R}^C$ is a learnable weight vector with elements close to 1 and $\mathscr{L}()$ a function that clips its input to the range $[0, 1]$. Notice that $\mathbf{a}_i^+$ also corresponds to the opacity value in volume rendering [26, 39]. Refer to the supplementary for examples of the learned primitives and convex shapes.

3.3 Differentiable rendering

We follow pixelNeRF [53] to compute the color for each query point. We use the same network specification, except we remove the volume density prediction branch. Specifically, given a pixel in an input view, we obtain the RGB color $\widehat{\mathbf{c}}$ of each query point along the corresponding ray using MLPs. The inputs include the 3D position, ray direction, and image features projected from all views.

By accumulating over all query points i along that ray, we render the color at the given pixel as: $\widehat{\mathbf{C}} = \sum_{i=1}^{R} t_i \alpha_i \widehat{\mathbf{c}}_i$, where R is the number of sampled points along the ray, α_i denotes the opacity which takes on different values at different phases of the optimization (see Sec. 3.4), and $t_i = \prod_{k=1}^{i-1}(1 - \alpha_k)$ is the accumulated transmittance. We also render the object mask value as $\widehat{\mathbf{M}} = \sum_{i=1}^{\Omega} t_i \alpha_i$.

3.4 Network training and test-time adaptation

Similar to [25, 39, 53], DPA-Net can work in a *feed-forward* fashion by training on a dataset including sparse-view images. We also propose a multi-phase test-time adaptation for instance-specific refinement during inference.

Pretraining: Our loss function consists of three terms:

$$\mathcal{L} = \mathcal{L}_{ph} + \mathcal{L}_{\mathbf{T}} + \mathcal{L}_{\mathbf{w}}, \tag{3}$$

$$\mathcal{L}_{ph} = \frac{1}{B}\sum_{j=1}^{B}\left\|\widehat{\mathbf{C}}_j - \mathbf{C}_j\right\|_2^2 + \frac{1}{B}\sum_{j=1}^{B}\left\|\widehat{\mathbf{M}}_j - \mathbf{M}_j\right\|_2^2, \tag{4}$$

$$\mathcal{L}_{\mathbf{T}} = \sum_{i,j}\max(-\mathbf{T}_{i,j}, 0) + \max(\mathbf{T}_{i,j} - 1, 0), \tag{5}$$

$$\mathcal{L}_{\mathbf{w}} = \sum_{j}|\mathbf{w}_j - 1|, \tag{6}$$

where $\mathcal{L}_{ph}$ including an abstraction loss and a mask loss enforces photo consistency, B is #pixels in a training batch, and $\mathbf{C}$ and $\mathbf{M}$ denote GT color and mask, respectively. We define the opacity $\alpha = \mathbf{a}^+$ from Eq.(2). $\mathcal{L}_{\mathbf{T}}$ keeps entries of the selection matrix $\mathbf{T}$ within range $[0, 1]$. $\mathcal{L}_{\mathbf{w}}$ ensures that entries of the weight vector $\mathbf{w}$ close to 1. All training instances share the same $\mathbf{T}$ and $\mathbf{w}$.

Test-time adaptation (TTA): We have found that DPA-Net trained on a generic dataset risks suboptimal performance when applied to novel shapes which deviate substantially from the training data. For example, assembling each 3D object requires unique matrices $\mathbf{T}$ and $\mathbf{w}$ tailored to that instance. To improve generalization, we propose a test-time adaptation that operates in three phases with different configurations as shown in Table 1. Each phase is initialized from its preceding phase.

Phase 1: The same as pretraining except that the input test images are used as GT.

Phase 2: Instead of using the approximated occupancy field $\mathbf{a}^+$, we opt to employ the exact definition $\mathbf{a}^*$ to further improve the abstraction performance. The loss function becomes $\mathcal{L}_{ph} + \mathcal{L}_{\mathbf{T}}$, whereas $\mathcal{L}_{\mathbf{w}}$ is omitted since $\mathbf{a}^*$ is devoid of $\mathbf{w}$ as in Eq.(1). We set the opacity $\alpha = \exp(-10\mathbf{a}^*)$.

Phase 3: Ideally, the selection matrix $\mathbf{T}$ should be binary to exactly mimic the operation of quadric primitive intersection. To ensure that, we follow [6, 55] to binarize $\mathbf{T}$ with a threshold of 0.01 and freeze it at this phase. We then fine-tune the network with two loss terms: the photo-consistency term $\mathcal{L}_{ph}$ and an additional regularization term $\mathcal{L}_{over}$ that discourages excessive overlap between convex shapes.

Overlapping loss: As observed in [29, 36], it is non-trivial to explicitly calculate and penalize area overlaps. Instead, we constrain points lying inside overlapped areas. Concretely, we define $\mathbf{h}_i = \sum_{c=1}^{C}\exp(-10\mathbf{O}_{i,c})$ for the ith point, where $\mathbf{h}_i$ is large if the point lies within multiple convexes. We minimize $\mathcal{L}_{over} = \frac{1}{\#\Omega}\sum_{i\in\Omega}\max(\mathbf{h}_i, 1.9)$, where Ω is the set of indexes of 3D points inside the shape, with cardinality $\#\Omega$. In practice, we sample 40,960 points surrounding the recovered shape in Phase 2 and utilize the condition $\mathbf{a}_i^* < 0.01$ to check point membership in Ω.

4 Network Improvements

The core modules of DPA-Net as described in Sec. 3 can still underperform when abstracting objects with complex structures. One issue comes from objects with holes and

Table 1: Configurations for multi-phase test-time adaption (TTA).

Phase	Type of $\mathbf{T}$	Type of $\mathbf{w}$	Occupancy	Opacity	Dropout	Loss	Model parameters
1	float	float	$\mathbf{a}^+$	$\mathbf{a}^+$	-	$\mathcal{L}_{ph} + \mathcal{L}_{\mathbf{T}} + \mathcal{L}_{\mathbf{w}}$	network, $\mathbf{T}$, $\mathbf{w}$
2	float	-	$\mathbf{a}^*$	$\exp(-10\mathbf{a}^*)$	-	$\mathcal{L}_{ph} + \mathcal{L}_{\mathbf{T}}$	network, $\mathbf{T}$
3	binary	-	$\mathbf{a}^*$	$\exp(-10\mathbf{a}^*)$	✓	$\mathcal{L}_{ph} + \mathcal{L}_{over}$	network

the other is redundant primitives. To address these issues, we present two key improvements to our network.

Silhouette-aware pixel sampling: By default, NeRF [26] and variants randomly sample pixels during optimization, which was to ensure a balanced learning. However, we found DPA-Net to struggle with it when abstracting 3D objects with complex topologies, *e.g.*, thin structures or holes. To address this, we adopt adaptive pixel sampling to allocate more pixels near *object silhouettes*. Specifically, we sample two pixel sets: 256 pixels randomly across the entire raster space, and 1,000 pixels around object contours. The contour pixels are obtained by performing morphological closing on the mask image and then adding Gaussian noise to the pixel coordinates.

Such an adaptive sampling scheme places more emphasis on boundary areas during optimization to improved abstraction amid complex 3D topologies. We believe that it is a *general* sampling strategy which could potentially benefit other abstraction frameworks other than DPA-Net. We leave this investigation to future work.

Primitive dropouts: Attaining a compact primitive assembly, with a low primitive count, without compromising abstraction accuracy is clearly desirable.

Similar to [54], we drop out a primitive if excluding it from the construction does not significantly alter the overall shape. In practice, we initialize with the primitives obtained from Phase 3. For each pth primitive, we set the pth row of the binary selection matrix $\mathbf{T}$ to zero, compute a new occupancy field $\mathbf{b}^*$ using Eq.1, and finally quantify the shape variation as: $v = \frac{1}{K}\sum_{i=1}^{K} \mathbb{1}\left(\text{XOR}(\mathbf{b}_i^* < 0.1, \mathbf{a}_i^* < 0.1)\right)$, where $K = 40,960$ is the number of points sampled around the shape as in Phase 3 and $\mathbb{1}()$ maps Boolean values to 0 or 1.

If v is close to zero, we keep the pth row at zero. Otherwise, we restore the original values. We iterate through each row of $\mathbf{T}$ to retain the set of primitives corresponding to non-zero rows, where this primitive dropout is applied at every 400 iterations during Phase-3 fine-tuning. Our experiments show that this strategy can remove up to 29% redundant primitives; see supplementary for more details.

5 Results & Evaluation

We implement DPA-Net in PyTorch [38], using the Adam [21] optimizer with a learning rate of 0.0001. Each batch contains 4 object instances. Our model is trained and evaluated on 4 NVIDIA Tesla V100 GPUs. We evaluate on synthetic images rendered from ShapeNet [3] and real images from the MVS DTU benchmark [15], using standard metrics for 3D abstraction quality and shape compactness. Pre-training on ShapeNet takes about three days. TTA on average takes one hour for ShapeNet and two hours for DTU.

We compare our method with two recent 3D shape abstraction methods, EMS [24] and DBW [29], for which open-source code is available[3]. In addition, we also compare with two reconstruction methods, pixelNeRF [53] and DiViNet [46].

EMS [24] is an optimization-based method for fitting super-quadrics on 3D points. We sample 6K points from the GT meshes and use parameter settings from [24].

DBW [29] is the SOTA method to recover super-quadric meshes from *dense-view* images. We adapted it for sparse views but failed to obtain meaningful results. For a fair comparison, we instead compare to it using intended dense image inputs on DTU.

pixelNeRF [53] is an image-conditioned NeRF for novel view synthesis from sparse images. In its original implementation, the volume density is predicted as a function of both location and viewing direction. However, the density value for each point should be independent of the viewing direction [26], which motivates us to modify its network architecture to learn better volume density field for subsequent geometry extraction. In particular, we predict the density using the first three ResBlocks that do not take the viewing direction as input. We then extract the final mesh by applying marching cubes on the inferred density field. Note that the output geometry is un-structured.

DiViNet [46] is the SOTA method for neural 3D surface reconstruction from sparse-*and* disparate-view RGB images, via volume rendering. It uses learned 3D Gaussians as shape template priors and "stitch" the shape surface to the template.

Metrics: We evaluate our method from three aspects. (1) 3D abstraction quality, measured by three standard metrics [6,55]: symmetric Chamfer Distance (CD), Edge Chamfer Distance (ECD), and Normal Consistency (NC). The same parameter settings from CAPRI-Net [55] are used to calculate CD and ECD. (2) Shape compactness, which is reflected by having fewer parts (#Parts) and leads to ease of manipulation/editing. (3) Rendering quality, measured by the standard image quality metrics PSNR and SSIM.

5.1 Evaluation on ShapeNet

We first evaluate on ShapeNet [3] with pre-rendered ShapeNet images from SRN [41] and Kato *et al*. [19]. We use GT camera poses for both training and testing, as well as object masks from pixelNeRF, which were obtained via thresholding.

Category-specific setting: In this setting, we evaluate the methods on a single chair category. We follow pixelNeRF and adopt the training/testing split used in SRN, with a total of 6,591 chairs. Each chair model is rendered from 50 different viewpoints to generate the training images. At training time, we randomly samples 3 views for each training step. At test time, we use 2 front-view images and 1 back-view image for each test chair. All images have 128×128 resolution. We compare DPA-Net against EMS [24] and pixelNeRF [53]. Table 2 presents the main quantitative results. EMS [24], while being a state-of-the-art primitive fitting methods from 3D inputs, produces larger errors across all metrics, while using a significantly larger number of 3D parts.

In particular, DPA-Net is better by 88%/17%/68% in CD/NC/ECD and uses 45% fewer 3D parts. Compared with the modified pixelNeRF, we obtain similar errors (differs by 0.3 in CD, 0.07 in NC, 0.8 in ECD, 0.55 in PSNR and 0.003 in SSIM) across

[3] Note that the code of ISCO [1] was not publicly available upon submission.

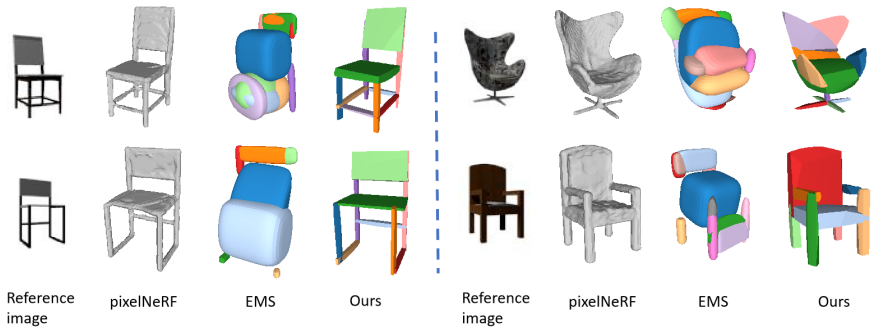

Fig. 3: Visual comparisons on the ShapeNet chair benchmark. The red ovals highlights noisy surfaces, which can be seen more clearly if the image is zoomed in upon.

Table 2: Quantitative comparisons against EMS [24] and pixelNeRF [53] on the ShapeNet chair benchmark. EMS takes 3D point cloud as input while pixelNeRF and our DPA-Net take 2D images. Since pixelNeRF outputs non-structured shapes, we mark with a "-" in the #Parts entry. The colors cyan and orange represent the best and the second best results.

Method	Input	CD↓	NC↑	ECD↓	#Parts↓	PSNR↑	SSIM↑
EMS	3D	6.93	0.65	12.56	14.92	-	-
pixelNeRF	2D	0.50	0.85	3.21	-	27.85	0.963
Ours	2D	0.79	0.78	4.01	8.15	27.30	0.960

all metrics. However, our method is more flexible in that it also produces convex part decomposition in an unsupervised manner, which is more useful in many applications, compared to the non-structured mesh produced by pixelNeRF.

The qualitative results in Fig. 3 consolidates our conclusion above. Both pixelNeRF and DPA-Net are able to accurately recover the object geometry. However, due to the non-structured 3D representation, pixelNeRF suffers from noisy surfaces (highlighted by the red ovals). EMS is able to create smooth surfaces but exhibits inferior 3D abstraction quality. In contrast, DPA-Net not only abstracts reasonably accurate 3D shape but also produces clean and meaningful shape structure decomposition.

Ablation studies: We validate the effectiveness of four important strategies on improving DPA-Net: test-time adaptation (TTA), adaptive pixel sampling (AS), overlapping loss (OL) and primitive dropout (PD). Table 3 shows the main quantitative results. Removing TTA (w/o TTA) and adaptive sampling (w/o AS) significantly degrades the abstraction quality as the CD increases by 54% and 41% respectively, validating the contribution of our TTA strategy and silhouette-aware pixel sampling method. Removing the overlapping loss (w/o OL) results in higher number of primitives and convex shapes, showing that this loss helps in removing redundant, heavily overlapped convexes. The primitive dropout (PD) strategy further reduces the number of primitives by 29% without sacrificing the abstraction accuracy.

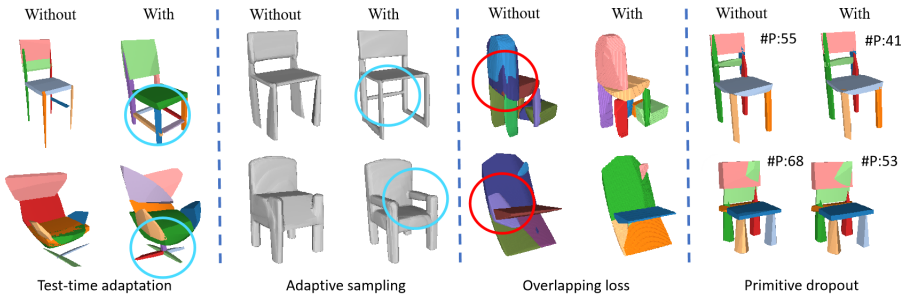

Fig. 4: Ablation studies on the three key strategies to boost DPA-Net. The texts, #P's, in the last column show the number of qudaric primitives used. The cyan ovals highlight challenging reconstruction areas that our method with TTA or adaptive sampling correctly recovers. The red ovals highlight redundant overlapping convex parts.

Table 3: Ablation studies on the ShapeNet chair benchmark. #P and #C denote the number of quadric primitives and convex shapes used during shape assembly. The colors cyan and orange represent the best and the second best results.

Method	CD ↓	NC ↑	ECD ↓	#P ↓	#C ↓
w/o TTA	1.22	0.81	6.11	39.04	7.38
w/o AS	1.11	0.75	5.31	45.81	8.23
w/o OL	0.83	0.82	4.03	61.23	8.61
w/o PD	0.81	0.78	3.85	59.64	8.45
Full model	0.79	0.78	4.01	42.38	8.15

Fig. 4 qualitatively demonstrates the contributions of the four designs. TTA helps recover missing structures from the pretraining results, while adaptive pixel sampling aids in capturing thin structures and achieving accurate shape silhouettes. Enforcing a overlap loss removes redundant 3D parts. Primitive dropouts effectively reduce the number of primitives for assembling the final shape.

Category-agnostic setting: For experiments in a *category-agnostic* way, we train our network on 13 largest categories of ShapeNet, covering roughly 43K object instances. We again follow pixelNeRF and adapt the training/testing split in Kato *et al*. [19], in which multi-view images, at resolution 64×64, are rendered from 24 fixed elevated angles. During training, we randomly pick 3 views from those 24 views for each object.

Table 4 summarizes the main results. DPA-Net consistently outperforms EMS across all metrics by a significant margin, except for NC, reinforcing our category-specific findings. DPA-Net being beat on NC is primarily due to the challenges posed by sparse view supervision. For instance, the table top in our results is not perfectly flat, which is a consequence of fine-tuning with a limited number of views. In contrast, EMS, which utilizes points from GT meshes, are more apt at fitting well-oriented primitives.

More importantly, while pixelNeRF attained higher accuracy when trained solely on chairs, our DPA-Net surpasses pixelNeRF across all metrics (except for NC and PSNR) in this diverse multi-category experiment — for instance, improving on the CD

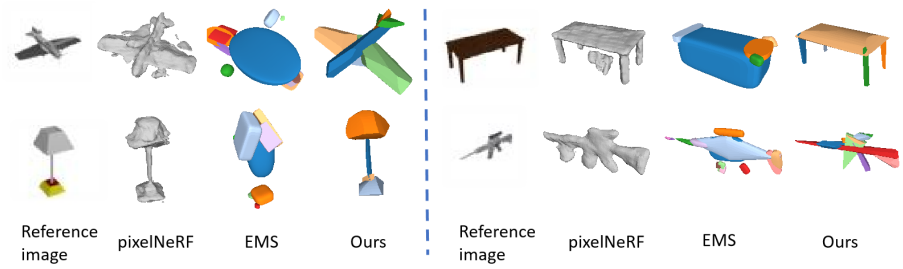

Fig. 5: Qualitative comparisons in the category-agnostic setting. The image resolution is 64×64.

Table 4: Quantitative evaluations in the category-agnostic setting on the ShapeNet dataset. Results marked by the color cyan are the best.

Method	Input	CD↓	NC↑	ECD↓	#Parts↓	PSNR↑	SSIM↑
EMS	3D	5.15	0.75	43.45	8.28	-	-
pixelNeRF	2D	3.67	0.73	11.01	-	27.17	0.923
Ours	2D	1.47	0.70	4.91	5.57	25.96	0.932

metric by nearly 60%. This highlights the enhanced capacity of our structured primitive-based representation in capturing expansive shape variations compared to pixelNeRF's unstructured approach.

As shown in Fig. 5, DPA-Net successfully recovers 3D primitive assemblies closely approximating GT shapes despite relatively low 64×64 input image resolution. In summary, our multi-category evaluations further validate the effectiveness DPA-Net in generalized shape abstraction and structured primitive discovery from limited visual data across substantially diverse object classes.

5.2 Evaluation on DTU

To further demonstrate the applicability of our method for learning primitive assemblies, we also test on the real images from the DTU dataset [15], which contains multi-view scene images with curated backgrounds captured in a controlled indoor environment. Each scene contains one or more objects. We use the camera poses and object masks from DVR [32] and resize all images to 300×400. Following standard practices, we treat the structured light scans as 3D GT. Instead of pre-training on DTU, we apply TTA directly on the 6 scenes with object masks used in DBW [29]. This challenging setting allows us to rigorously test the generalizability of DPA-Net to real images.

As shown in Table 5, DPA-Net obtains higher-quality 3D abstractions with 18% lower average CD while using 3.5 more parts compared to DBW. DiViNet [46] and pixelNeRF [53] outperform DAP-Net as they output un-structured meshes. Fig. 6 provides visual comparisons showing that our method can recover fine structural details.

5.3 Applications

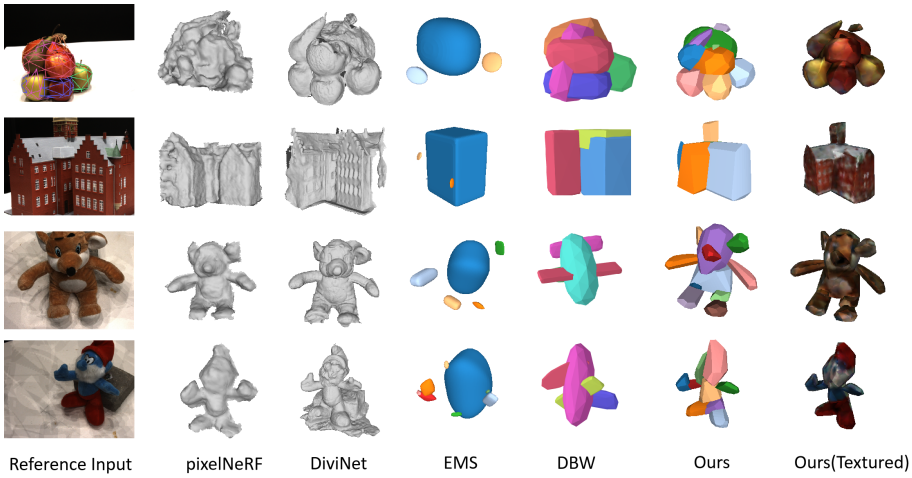

Fig. 6: Visual comparisons on 3D abstraction and reconstruction over the DTU dataset.

Table 5: Quantitative comparisons on 6 DTU scenes with object masks. As DiViNet [46] outputs un-structured shapes, we mark with a "-" in the #Parts entry. Cyan marks the best results.

Method	Input	Structure	S24	S40	S55	S63	S83	S105	Mean CD	Mean #Parts
EMS	3D	✓	4.04	5.85	4.13	4.99	3.54	5.99	4.75	4.50
DBW	2D	✓	3.85	1.34	3.83	4.57	5.05	6.07	4.12	3.83
Ours	2D	✓	4.74	3.27	2.88	2.99	3.26	3.30	3.41	7.33
pixelNeRF	2D	×	3.54	2.07	2.27	4.57	3.52	2.12	3.01	-
DiViNet	2D	×	3.34	1.47	0.75	2.74	1.94	1.17	1.90	-

The structured 3D abstractions obtained by DPA-Net are fully interpretable since they represent primitives assembled by meaningful CAD operations. This allows the obtained results to be directly edited or animated for down-streaming applications. Users can easily modify the abstracted parts by adding, removing, translating, or rotating components to create desired CAD designs, as demonstrated in Figs. 1 and 7. These 3D shape editing results were obtained by MeshLab [7].

In addition, we can fit basic shapes (e.g., cuboids) to the abstracted convex shapes and export them to an OpenSCAD script, enabling programmatic editing of 3D shapes in CAD software [42], as demonstrated in the top row in Fig. 8.

Last but not least, the abstracted 3D shapes obtained by DPA-Net can serve as easy-to-edit "structural prompts" [5, 40] for novel and detailed 3D shape generations. Here, we take the 3D generation method from SPiC-E [40] as an example. SPiC·E can transform primitive-based abstractions into highly expressive shapes according to text conditions. Taking one result from DPA-Net, for example, we produce three edited shapes from OpenSCAD as input guidance 3D shape, which are referred to as structural prompts in Fig. 8. We input these guidance 3D shapes and different text conditions into

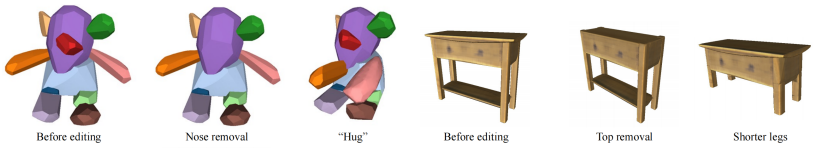

Fig. 7: Direct editing, using MeshLab [7], over structured abstractions obtained by DPA-Net.

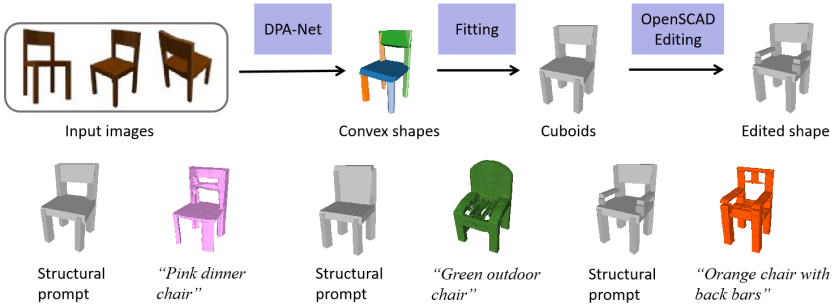

Fig. 8: Top: editing 3D shape from DPA-Net by OpenSCAD [42]. Bottom: 3D shape generation by SPiC-E [40] from text prompts and 3D guidance shapes as structural prompts. The generated detailed shapes share a similar structure as the input 3D guidance shapes and have more details according to the text condition.

SPiC-E and generate more detailed shapes. The results demonstrate that our method holds strong application potential for such promising downstream tasks.

6 Conclusion

We present DPA-Net, a differentiable framework for learning structured 3D abstractions in the form of primitive assemblies from only few, e.g., three, RGB images captured at very different views. Our key innovation is the integration of differential primitive assembly into the NeRF architecture, enabling the prediction of occupancies to serve as opacity values for volume rendering. Without any 3D or shape decomposition supervision, our method can produce an interpretable, and subsequently editable, union of convexes that approximates the target 3D object. Quantitative and qualitative evaluations on ShapeNet and DTU demonstrate superiority of DPA-Net over state-of-the-art alternatives. The demonstrated applications further show that our editable 3D abstractions can serve as structural prompts and benefit other 3D generation tasks.

Our current implementation makes use of GT camera poses. To alleviate performance degradations caused by estimated, noisy poses, existing methods for joint camera-scene optimization, e.g., [44], may be applied. As texture prediction is not a focus of our work, further fine-tuning (e.g., with a bias towards input views) and optimization are needed to improve rendering quality. Finally, assembly using only convexes is limiting. As shown in the supplementary, DPA-Net does not handle concave shapes well. Adding difference operations into the differentiable assembly is worth exploring.

Acknowledgement

We thank all the anonymous reviewers and area chairs for their constructive comments. We also thank Kai Wang and Jianbo Ye both from Amazon for proofreading.

References

1. Alaniz, S., Mancini, M., Akata, Z.: Iterative superquadric recomposition of 3d objects from multiple views. In: ICCV (2023)
2. Cai, S., Obukhov, A., Dai, D., Gool, L.V.: Pix2NeRF: Unsupervised conditional π-gan for single image to neural radiance fields translation. In: CVPR (2022)
3. 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. arXiv (2015)
4. Chaudhuri, S., Ritchie, D., Wu, J., Xu, K., Zhang, H.: Learning generative models of 3d structures. Computer Graphics Forum (Eurographics STAR) (2020)
5. Chen, Q., Chen, Z., Zhou, H., Zhang, H.: Shaddr: Real-time example-based geometry and texture generation via 3d shape detailization and differentiable rendering. arXiv preprint arXiv:2306.04889 (2023)
6. Chen, Z., Tagliasacchi, A., Zhang, H.: BSP-Net: Generating compact meshes via binary space partitioning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 45–54 (2020)
7. Cignoni, P., Callieri, M., Corsini, M., Dellepiane, M., Ganovelli, F., Ranzuglia, G.: MeshLab: an Open-Source Mesh Processing Tool. In: Scarano, V., Chiara, R.D., Erra, U. (eds.) Eurographics Italian Chapter Conference. The Eurographics Association (2008). https://doi.org/10.2312/LocalChapterEvents/ItalChap/ItalianChapConf2008/129-136
8. Daxuan Ren, J.Z., Cai, J., Jiatong Li, H.J., Cai, Z., Zhang, J., Pan, L., Zhang, M., Zhao, H., Yi, S.: CSG-Stump: A learning friendly csg-like representation for interpretable shape parsing. In: ICCV. pp. 12458–12467 (2021)
9. Deng, B., Genova, K., Yazdani, S., Bouaziz, S., Hinton, G., Tagliasacchi, A.: CvxNet: Learnable convex decomposition. In: CVPR (2020)
10. Deng, K., Liu, A., Zhu, J.Y., Ramanan, D.: Depth-Supervised NeRF: Fewer views and faster training for free. In: CVPR (2022)
11. Ganeshan, A., Jones, R.K., Ritchie, D.: Improving unsupervised visual program inference with code rewriting families. In: ICCV (2023)
12. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
13. Huang, Q., Wang, H., Koltun, V.: Single-view reconstruction via joint analysis of image and shape collections. ACM TOG **34**(4) (2015)
14. Izadinia, H., Shan, Q., Seitz, S.M.: IM2CAD. In: CVPR (2017)
15. Jensen, R., Dahl, A., Vogiatzis, G., Tola, E., Aanæs, H.: Large scale multi-view stereopsis evaluation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 406–413 (2014)
16. Jiang, H., Jiang, Z., Grauman, K., Zhu, Y.: Few-view object reconstruction with unknown categories and camera poses. arXiv preprint arXiv:2212.04492 (2022)
17. Jones, R.K., Walke, H., Ritchie, D.: PLAD: Learning to infer shape programs with pseudo-labels and approximate distributions. In: CVPR (2022)

18. Kania, K., Zieba, M., Kajdanowicz, T.: Ucsg-net-unsupervised discovering of constructive solid geometry tree. Advances in Neural Information Processing Systems **33**, 8776–8786 (2020)
19. Kato, H., Ushiku, Y., Harada, T.: Neural 3d mesh renderer. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 3907–3916 (2018)
20. Kerbl, B., Kopanas, G., Leimkühler, T., Drettakis, G.: 3D gaussian splatting for real-time radiance field rendering. ACM Trans. on Graphics (SIGGRAPH) (2023)
21. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
22. Li, P., Guo, J., Zhang, X., ming Yan, D.: SECAD-Net: Self-supervised CAD reconstruction by learning sketch-extrude operations. In: CVPR (2023)
23. Liu, R., Wu, R., Hoorick, B.V., Tokmakov, P., Zakharov, S., Vondrick, C.: Zero-1-to-3: Zero-shot one image to 3d object. In: CVPR (2023)
24. Liu, W., Wu, Y., Ruan, S., Chirikjian, G.S.: Robust and accurate superquadric recovery: A probabilistic approach. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2676–2685 (2022)
25. Long, X., Lin, C., Wang, P., Komura, T., Wang, W.: SparseNeuS: Fast generalizable neural surface reconstruction from sparse views. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII. pp. 210–227. Springer (2022)
26. Mildenhall, B., Srinivasan, P.P., Tancik, M., Barron, J.T., Ramamoorthi, R., Ng, R.: NeRF: Representing scenes as neural radiance fields for view synthesis. Communications of the ACM **65**(1), 99–106 (2021)
27. Mitra, N., Wand, M., Zhang, H., Cohen-Or, D., Bokeloh, M.: Structure-aware shape processing. In: Eurographics State-of-the-art Report (STAR) (2013)
28. Mo, K., Guerrero, P., Yi, L., Su, H., Wonka, P., Mitra, N., Guibas, L.: StructureNet: Hierarchical graph networks for 3d shape generation. ACM TOG **38**(6) (2019)
29. Monnier, T., Austin, J., Kanazawa, A., Efros, A.A., Aubry, M.: Differentiable Blocks World: Qualitative 3D Decomposition by Rendering Primitives. In: NeurIPS (2023)
30. Müller, T., Evans, A., Schied, C., Keller, A.: Instant neural graphics primitives with a multiresolution hash encoding. ACM Transactions on Graphics (ToG) **41**(4), 1–15 (2022)
31. Niemeyer, M., Barron, J.T., Mildenhall, B., Sajjadi, M.S.M., Geiger, A., Radwan, N.: RegNeRF: Regularizing neural radiance fields for view synthesis from sparse inputs. In: CVPR (2022)
32. Niemeyer, M., Mescheder, L., Oechsle, M., Geiger, A.: Differentiable volumetric rendering: Learning implicit 3d representations without 3d supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3504–3515 (2020)
33. Niu, C., Li, J., Xu, K.: Im2Struct: Recovering 3D shape structure from a single RGB image. In: CVPR (2018)
34. Niu, C., Li, M., Xu, K., Zhang, H.: Rim-net: Recursive implicit fields for unsupervised learning of hierarchical shape structures. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11779–11788 (2022)
35. Paschalidou, D., Gool, L.V., Geiger, A.: Learning unsupervised hierarchical part decomposition of 3d objects from a single rgb image. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1060–1070 (2020)
36. Paschalidou, D., Katharopoulos, A., Geiger, A., Fidler, S.: Neural parts: Learning expressive 3d shape abstractions with invertible neural networks. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 3204–3215 (2021)
37. Paschalidou1, D., Ulusoy, A.O., Geiger, A.: Superquadrics revisited: Learning 3D shape parsing beyond cuboids. In: CVPR (2019)

38. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems **32** (2019)
39. Ren, Y., Zhang, T., Pollefeys, M., Süsstrunk, S., Wang, F.: Volrecon: Volume rendering of signed ray distance functions for generalizable multi-view reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 16685–16695 (2023)
40. Sella, E., Fiebelman, G., Atia, N., Averbuch-Elor, H.: Spic-e: Structural priors in 3d diffusion models using cross entity attention. arXiv preprint arXiv:2311.17834 (2023)
41. Sitzmann, V., Zollhöfer, M., Wetzstein, G.: Scene representation networks: Continuous 3d-structure-aware neural scene representations. In: Advances in Neural Information Processing Systems (2019)
42. openscad team: Openscad: The programmers solid 3d cad modeller. `https://openscad.org/`
43. Tertikas, K., Paschalidou, D., Pan, B., Park, J.J., Uy, M.A., Emiris, I., Avrithis, Y., Guibas, L.: Generating part-aware editable 3d shapes without 3d supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4466–4478 (2023)
44. Truong, P., Rakotosaona, M.J., Manhardt, F., Tombari, F.: SPARF: Neural radiance fields from sparse and noisy poses. In: CVPR (2023)
45. Tulsiani, S., Su, H., Guibas, L.J., Efros, A.A., Malik, J.: Learning shape abstractions by assembling volumetric primitives. In: CVPR (2017)
46. Vora, A., Patil, A.G., Zhang, H.: DiViNeT: 3D reconstruction from disparate views via neural template regularization. In: NeurIPS (2023)
47. Wang, Q., Wang, Z., Genova, K., Srinivasan, P., Zhou, H., Barron, J.T., Martin-Brualla, R., Snavely, N., Funkhouser, T.: IBRNet: Learning multi-view image-based rendering. In: CVPR (2021)
48. Xie, Y., Takikawa, T., Saito, S., Litany, O., Yan, S., Khan, N., Tombari, F., Tompkin, J., Sitzmann, V., Sridhar, S.: Neural fields in visual computing and beyond. Comput. Graph. Forum (2022)
49. Xu, K., Zheng, H., Zhang, H., Cohen-Or, D., Liu, L., Xiong, Y.: Photo-inspired model-driven 3d object modeling. ACM Transactions on Graphics (Special Issue of SIGGRAPH) **30**(4), 80:1–80:10 (2011)
50. Xu, K., Chen, K., Fu, H., Sun, W.L., Hu, S.M.: Sketch2Scene: Sketch-based co-retrieval and co-placement of 3D models. ACM TOG p. Article 123 (2013)
51. Yang, J., Pavone, M., Wang, Y.: FreeNeRF: Improving few-shot neural rendering with free frequency regularization. In: CVPR (2023)
52. Yang, Z., Ren, Z., Bautista, M.A., Zhang, Z., Shan, Q., Huang, Q.: Fvor: Robust joint shape and pose optimization for few-view object reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 2497–2507 (2022)
53. Yu, A., Ye, V., Tancik, M., Kanazawa, A.: PixelNeRF: Neural radiance fields from one or few images. In: CVPR (2021)
54. Yu, F., Chen, Q., Tanveer, M., Mahdavi-Amiri, A., Zhang, H.: D^2CSG: Unsupervised learning of compact csg trees with dual complements and dropouts. In: NeurIPS (2023)
55. Yu, F., Chen, Z., Li, M., Sanghi, A., Shayani, H., Mahdavi-Amiri, A., Zhang, H.: CAPRI-Net: Learning compact CAD shapes with adaptive primitive assembly. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 11768–11778 (2022)
56. Yu, Z., Peng, S., Niemeyer, M., Sattler, T., Geiger, A.: Monosdf: Exploring monocular geometric cues for neural implicit surface reconstruction. arXiv preprint arXiv:2206.00665 (2022)