Improved Convex Decomposition with Ensembling and Negative Primitives

Vaibhav Vavilala[1] Florian Kluger[2] Seemandhar Jain[1,3] Bodo Rosenhahn[2]
Anand Bhattad[4] David Forsyth[1]

[1]University of Illinois Urbana-Champaign [2]Leibniz Universität Hannover
[3]University of California, San Diego [4]Johns Hopkins University

Abstract

Describing a scene in terms of primitives – geometrically simple shapes that offer a parsimonious but accurate abstraction of structure – is an established and difficult fitting problem. Different scenes require different numbers of primitives, and these primitives interact strongly. Existing methods are evaluated by comparing predicted depth, normals, and segmentation against ground truth. The state of the art method involves a learned regression procedure to predict a start point consisting of a fixed number of primitives, followed by a descent method to refine the geometry and remove redundant primitives.

CSG (Constructive Solid Geometry) representations are significantly enhanced by a set-differencing operation. Our representation incorporates negative *primitives, which are differenced from the positive primitives. These notably enrich the geometry that the model can encode, while complicating the fitting problem. This paper presents a method that can (a) incorporate these negative primitives and (b) choose the overall number of positive and negative primitives by ensembling. Extensive experiments on the standard NYUv2 dataset confirm that (a) this approach results in substantial improvements in depth representation and segmentation over SOTA and (b) negative primitives improve fitting accuracy. Our method is robustly applicable across datasets: in a first, we evaluate primitive prediction for LAION images.*

1. Introduction

Compact yet accurate representations of scene and object geometry have a wide range of applications, including image editing, motion planning, and grasping. In image editing, a compact representation makes it easy for a user to select what must be edited (e.g. [3, 63]), and accuracy is required for tasks like camera movement (Figure 8). In motion planning and grasping, compactness supports efficient computation and accuracy is required to produce solutions that work

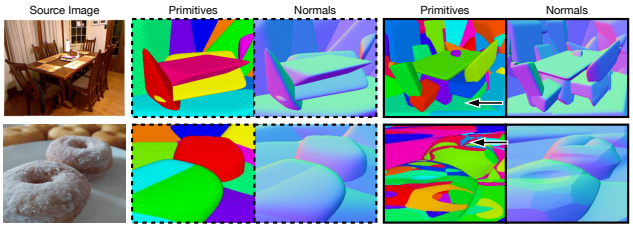


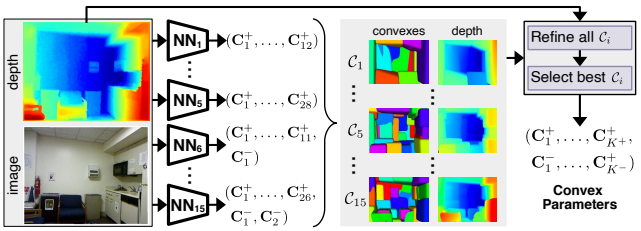
Figure 1. We present a method that can extract 3D primitives from RGB images with high geometric accuracy. Our method exploits negative primitives, which carve away geometry (i.e. a boolean operation). These subtractive operations enrich the shapes we can encode, such as holes in the donuts (**bottom row**). While no GT labels exist for the primitive parameters, we show how test-time ensembling can be used to find the appropriate level of abstraction for a given image. The second pair of results from each row can be obtained via ensembling, which produces more accurate geometric fits. Our test-time ensembling involves running several independent models and selecting the best solution for that image.

(e.g. [14, 37, 42, 58, 64, 68]. This paper describes a method that parses a scene into a set of simple, canonical shapes, or primitives. Our method balances three criteria: (i) *compactness* – the scene is represented with few primitives; (ii) *accuracy* – the geometric error between the representation and the ground truth scene is small; and (iii) *semantic coherence* – primitive boundaries are approximately aligned with object or part boundaries.

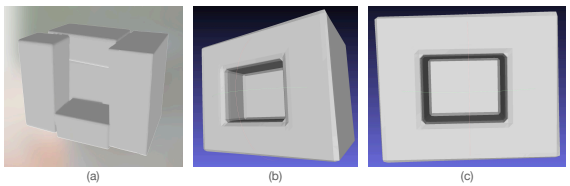
We follow [9] in using smoothed 3D polytopes, which avoid the geometric difficulties (e.g. singularities, non-convexities, self-intersections and symmetries) associated with superquadrics. Existing approaches to fit primitives are either descent methods (which optimize primitive parameters directly, and so susceptible to poor initialization and local minima) or regression methods (which predict parameters in a single pass, so can avoid the worst local minima but lack the precision to produce a tight fit to the specific input geometry). In existing methods, model selection (choosing the number of primitives for a scene) is by a single hyperpa-

rameter. But some scenes are more complicated than others (Fig. 1 top row). Further, existing methods lack the expressiveness to model concavities or holes efficiently (Fig. 1 bottom row). In contrast to [9], we use a hybrid regress then descent method; fit entire scenes using RGB input on a very large scale; fit negative primitives, allowing set differencing to make concavities; choose geometric complexity of the fitted representation with an adaptive method built on ensembling; and significantly exceed SOTA accuracy with very small representations. Our contributions are:

- **Fitting CSG Models with Negative Primitives**: We present the first method capable of fitting CSG models that include a set differencing operator to in-the-wild images of scenes. We demonstrate significant quantitative and qualitative benefits to using negative primitives for scene decomposition.
- **Test-Time Ensembling for Model Selection**: We introduce a test-time search procedure that dynamically selects the number of positive and negative primitives for each scene. This data-driven approach yields a more accurate representation than any fixed-count model.
- **State-of-the-Art Accuracy:** Our combined method significantly outperforms all available baselines and prior state-of-the-art on the NYUv2 benchmark, achieving a relative error reduction of over 50% and demonstrating strong generalization to diverse, in-the-wild scenes.
- **Viability at large scale:** We demonstrate fitting to over 1M in-the-wild LAION images with high accuracy.



(a) **Inference Overview:** An RGBD image is input to an ensemble of independently trained CNNs. Each network predicts the parameters of a set of convexes $\mathcal{C}_i$. The number of convexes varies between 12 and 36 in this work, with many of them potentially being negative. We refine each set of convexes by minimizing the training loss w.r.t. the input depth map. Our final decomposition consists of the set of refined convexes $\mathcal{C}_i$ which yields the lowest absolute relative depth error.



(b) **Negative primitives are parameter-efficient.** Representing a simple box with a hole punched in it can be challenging even with several traditional primitives, as shown in **(a)**, where five primitives get stuck in a local minimum. In contrast, two primitives - one positive and one negative - can represent the geometry successfully because of the enriched vocabulary of operations. Two views are shown in **(b)** and **(c)**.

Figure 2. Overview of our approach (left) and demonstration of negative primitives (right).

2. Related Work

Primitives date to the origins of computer vision. Roberts worked with blocks [50]; Binford with generalized cylinders [5]; Biederman with geons [4]. Ideally, complex objects might be handled with simple primitives [7] where each primitive is a semantic part [4, 5, 61]. Primitives can be recovered from image data [41, 52], and allow simplified geometric reasoning [45].

For individual objects, neural methods could predict the right set of primitives by predicting solutions for test data that are "like" those that worked for training data. Tulsiani *et al.* parse 3D shapes into cuboids, trained without ground truth segmentations [59]. Zou *et al.* parse with a recurrent architecture [74]. Liu *et al.* produce detailed reconstructions of objects in indoor scenes, but do not attempt parsimonious abstraction [34]. Worryingly, 3D reconstruction networks might rely on object semantics [56]. Deng *et al.* (CVXNet) represent objects as a union of convexes, again training without ground truth segmentations [9]. An early variant of CVXNet can recover 3D representations of poses from single images, with reasonable parses into parts [8]. Meshes can be decomposed into near convex primitives, by a form of search [66]. Part decompositions have attractive editability [20]. Regression methods face some difficulty producing different numbers of primitives per scene (CVXNet uses a fixed number; [59] predicts the probability a primitive is present; one also might use Gumbel softmax [24]). Primitives that have been explored include: cuboids [6, 13, 29, 38, 49, 53, 55, 59]; superquadrics [2, 23, 43]; planes [7, 32]; and generalized cylinders [31, 41, 72]. There is a recent review in [12].

Neural Parts [44] decomposes an object given by an image into a set of non-convex shapes. CAPRI-Net [71] decomposes 3D objects given as point clouds or voxel grids into assemblies of quadric surfaces. DeepCAD [67] decomposes an object into a sequence of commands describing a CAD model, but requires appropriately annotated data for training. Point2Cyl [60] is similar, but predicts the 2D shapes as an SDF. Notably, [60, 67, 71] also utilize CSG with negative parts but, unlike our work, focus on CAD models of single objects instead of complex real-world scenes.

Hoiem *et al.* parse outdoor scenes into vertical and horizontal surfaces [21, 22]; Gupta *et al.* demonstrate a parse into blocks [15]. Indoor scenes can be parsed into: a cuboid [17, 62]; beds and some furniture as boxes [18]; free space [19]; and plane layouts [33, 54]. If RGBD is available, one can recover layout in detail [73]. Patch-like primitives can be imputed from data [11]. Jiang demonstrates parsing

RGBD images into primitives by solving a 0-1 quadratic program [25]. Like that work, we evaluate segmentation by primitives (see [25], p. 12), but we use original NYUv2 labels instead of the drastically simplified ones in the prior work. Also, our primitives are truly convex. Monnier *et al.* and Alaniz *et al.* decompose scenes into sets of superquadrics using differentiable rendering, which requires calibrated multi-view images as input [1, 39]. Most similar to our work is that of Kluger *et al.*, who identify cuboids sequentially with a RANSAC-like algorithm [10, 27–30].

The success of a descent method depends critically on the start point, typically dealt with using greedy algorithms (rooted in RANSAC [10]; note the prevalence of RANSAC in a recent review [26]); randomized search [16, 46]; or multiple starts. Remarkably, as Sec. 3.2 shows, modern first-order methods are capable of producing fairly good primitive representations from a random start point [35].

3. Method

Write K^{total} for the total number of primitives and K^- for the number of negatives to be predicted. For each (K^{total}, K^-) we wish to investigate, we train a prediction network. Then, at inference time, we produce a set of primitives from each network, polish it, select the primitives with the best loss and report that, so different scenes will have predictions involving different numbers of primitives. Our approach generalizes the architecture of [62], but produces very significant improvements in performance (Sec. 4). Negative primitives require some minor modifications of their procedure (Sec. 3). Our losses (Sec. 3.1) and test-time refinement (Sec. 3.2) are somewhat similar.

Our network requires RGBD input. If depth measurements are not available, our method works well with single image depth predictors [48, 69]. Our losses require a point cloud that is extracted from the depth image via the heuristic described in [62]. Fig. 2a provides an overview of our inference pipeline.

Base primitives: Our primitives are smoothed polytopes as described in [9]. For 6-faced parallelepipeds, each primitive is parametrized by a center (3 DOF's), 3 normals (6 DOF's), 6 offsets (6 DOFs) and a blending term (1 DOF). The blended half-plane approach eases training and also enables fitting curved surfaces. We fit 6-faced parallelepipeds for fair comparative evaluation, and then show that more faces per polytope yields better representations (see Table 6 in supplementary).

Negative primitives: Set differencing produces a notably more complex geometric representation. Assume we have K^{total} primitives of which K^- are negative, each with f faces. Label an image pixel by the face intersection that produced that pixel (as in our face segmentation figures, e.g. Fig. 4). Generic pixels could result from either ray intersection with a face of a positive primitive or with a face of a negative primitive inside some positive. This argument means that there are a maximum of $f \times (K^{total} - K^-) \times (1 + K^-)$ pixel labels; note how this number grows very quickly with an increase in the number of negative primitives, an effect that can be seen in Fig. 4. Negative primitives are easily handled with indicator functions. We define the indicator for a set of primitives $O : \mathbb{R}^3 \rightarrow [0, 1]$, with $O(x) = 0$ indicating free space, and $O(x) = 1$ indicating a query point $x \in \mathbb{R}^3$ is inside the volume. Write $O^+(x)$ for the indicator function for the set of positive primitives, $O^-(x)$ for negative primitives. The indicator for our representation is then:

$$O(x) = relu(O^+(x) - O^-(x)). \quad (1)$$

As we will show in our results, Section 4, negative primitives allow us to encode a richer set of shapes, given the same primitive budget. Our analysis leads to the following theorem, which we analyze in the supplement Sec. 8:

Theorem 1. *For the vocabulary of primitives described, there exist shapes S such that $K_{\pm}(S) \ll K_{+}(S)$ when the representation of S is required to achieve sufficiently high precision.*

3.1. Losses

Our modified representation allows re-using the existing sample loss and auxiliary losses (unique parametrization loss, overlap loss, guidance loss, localization loss) [9, 62] for both $O^+(x)$ and $O^-(x)$. While a Manhattan World loss was found to be helpful for NYUv2, it hurt quality on general in-the-wild LAION images in our testing so we exclude it. We do not consider the volume loss or segmentation loss from [62] in our experimentation, as they were shown to have an approximately neutral effect in the original paper.

3.2. Polishing and Descent

Test-time finetuning is possible because we can evaluate the primitive prediction against the predicted depth map, then use the training losses at test-time. The fit is improved by using more 3D samples in these losses per image at test-time. Our polishing procedure is heavily optimized.

In fact, our test-time refinement procedure is effective enough that we can fit a primitive representation *using only a random start*. We are aware of no other primitive fitting procedure that can operate with pure descent and no random restart or incremental process. This supplies an interesting baseline; Sec. 4 demonstrates that this baseline is highly inefficient compared to polishing a network prediction, and is not competitive in accuracy.

3.3. Choosing the Number of Primitives

Much of the literature on primitive decomposition fits a fixed number of primitives [9]. In contrast, we train 18 models for (K^{total}, K^-), with $K^{total} \in \{\mathbf{12}, \mathbf{24}, \mathbf{36}\}$ and $K^- \in$

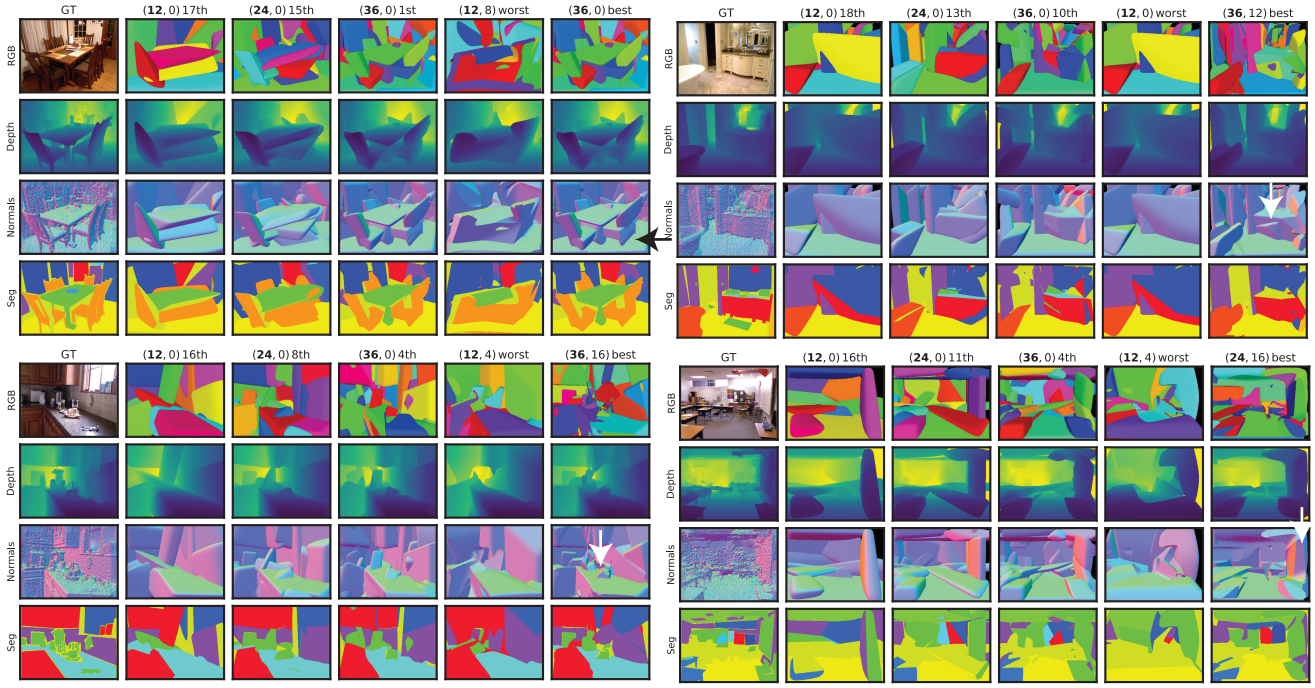


Figure 3. Visualizations of various primitive predictions for four scenes from NYUv2. We show ground truth (first column in each block); predictions of $(\mathbf{12}, 0)$, $(\mathbf{24}, 0)$ and $(\mathbf{36}, 0)$ models; the prediction of the model that made the *worst* prediction for the scene; and the prediction of the model that made the *best* prediction. The best choice of primitive counts K^{total} varies from scene to scene. Notice some complex objects made up as composites of positive primitives (black arrow) and negative primitives "carving out" shapes. The segmentation label is the oracle label described in the text.

$\{0, 4, ..., K^{total} - 4\}$. We investigate two strategies for choosing the best prediction (and so the best set of primitives) for a given test image: $S \rightarrow R$, where we select the best neural prediction then refine it; and $R \rightarrow S$, where we refine all predictions then select the best.

3.4. Implementation Details

Our neural architecture is a ResNet-18 encoder (first layer expanded to accept RGBD input), followed by a decoder consisting of three linear layers of sizes $[1048, 1048, 2048]$ and LeakyRelu activations. We do not freeze any layers during training. The dimensionality of the final output varies based on the number of primitives the model is trained to produce (as we train different models for different numbers of primitives in this work). We implement our procedure in PyTorch and train all networks with AdamW optimizer, learning rate 2×10^{-4}, batch size 96, mixed-precision training, for 20000 iterations, on a single A40 GPU. Each image is resized to 240×320 resolution. Although we train at fixed resolution, our model can run inference at variable aspect ratio, as would be expected from CNNs like ResNet. It takes 39 mins to train a 12 primitive model and 62 mins to train a 36 primitive model. On LAION, we train at 256×256 resolution. We use random flip, scale, and crop during training.

4. Results

Qualitative results appear in Fig 3 and Fig 4. Note how primitives can combine to form complex structures; how negative primitives "carve out" complex shapes; how primitives tend to "stick" to object properties (for example, heads; a house); and how the number of face labels grows very quickly with the number of negative primitives.

4.1. Evaluating a Primitive Representation

While *producing* a primitive representation has a long history ([36], Sec. 2), not much is known about how one is to be *used* apart from the original recognition argument, now clearly an anachronism. Recent work in conditioned image synthesis ([3, 63]) suggests that applications might need (a) a relatively compact representation (so that users can, say, move primitives around) and (b) one that accurately reflects depth, normals and (ideally) segmentation.

We compare primitive methods against one another using standard metrics for depth, normal and segmentation. Specialized predictors of depth, normal and segmentation outperform primitive methods on these metrics. But we would not use a primitive predictor to actually predict depth, normal or segmentation – instead, we are using the metrics

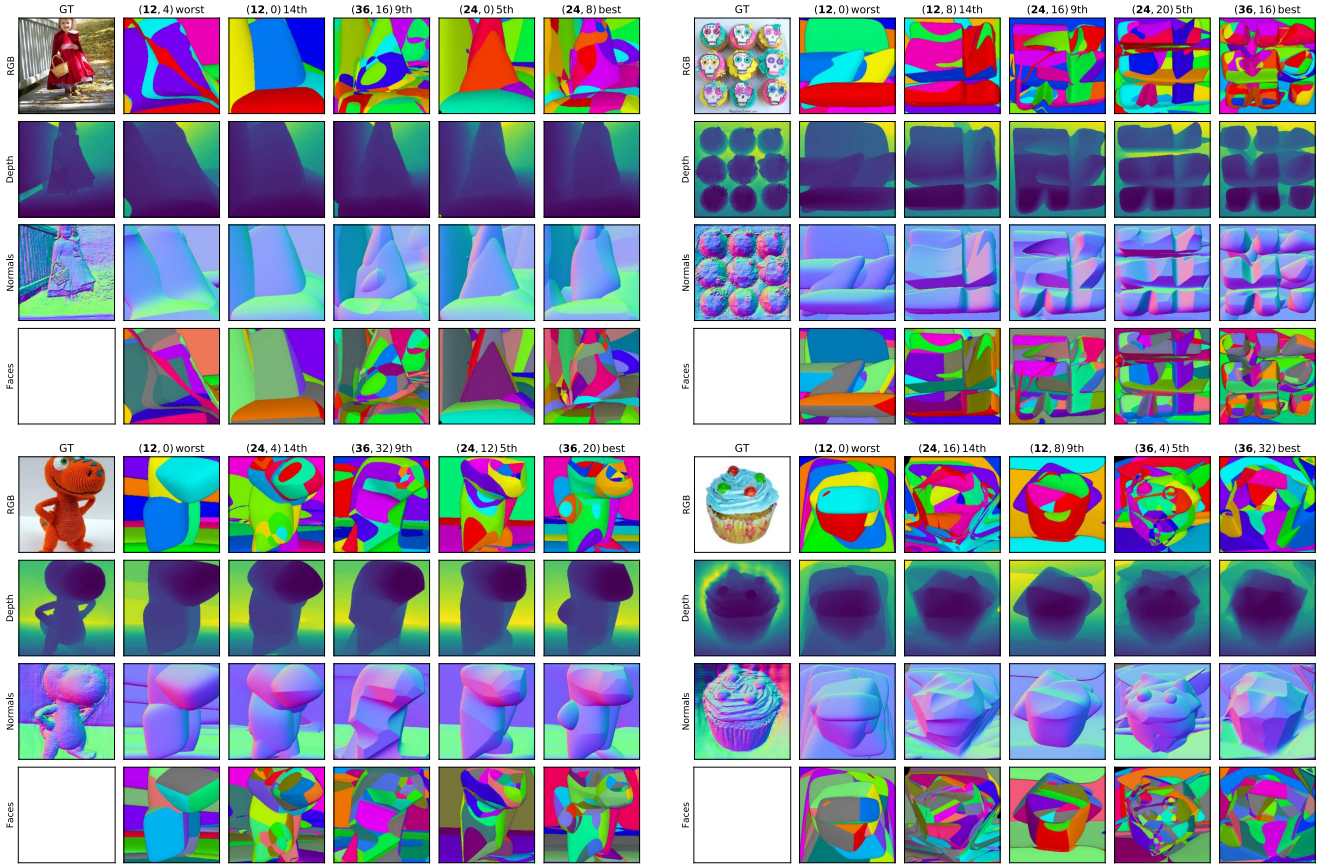


Figure 4. Visualizations of various primitive predictions for four scenes from LAION. We show ground truth (first column in each block); the prediction of the model that made the *worst* prediction for the scene; the prediction of the model that made the *best* prediction rightmost; and in-between results shown in intermediate columns. While our models produce good fits on average, ensembling helps select the optimal primitive count for each test scene and avoids poor solutions. The best choice of primitive counts K^{total} varies from scene to scene. **Bottom row** in each block shows face labels – no oracle segmentation is available. Note how primitives can follow complex structures; how they tend to "stick" to object properties; and how the number of face labels grows very quickly with the number of negative primitives. See Figs. 10 and 11 for additional examples.

to determine whether very highly simplified representations achieve reasonable accuracy. Our procedure uses the standard 795/654 train/test NYUv2 split [40]. We hold out 5% of training images for validation. We use this dataset primarily to maintain consistency in evaluating against prior art.

For NYUv2, we compare the depth map predicted by primitives to ground truth using a variety of metrics; normals predicted by primitives to ground truth; and a segmentation derived from primitives to ground truth segmentation. Depth metrics are: the (standard) AbsRel (eg [47]); AUC_n, which evaluates the fraction of points within n cm of the correct location (after [29, 62]); mean and median of the occlusion-aware distance of [29]. Normal metrics are based on [65] and are mean and median of angle to true normal, in degrees. The segmentation metric uses the GT segmentation to assign the best label for each image region, where regions consist of pixels with the same face intersection label (of Sec. 3), then compares this to ground truth. The SegAcc error metric shown in Tables 1 and 7 refers to the fraction of pixels labeled correctly. For LAION, we compute depth and normal metrics from generated primitives comparing to depth and normal predicted from the image.

4.2. NYUv2 Results

Our method beats SOTA on depth, normal, and segmentation (Table 1). With or without ensembling process, **our method is faster than SOTA**. Qualitative results in Fig. 3. More extensive detailed comparison in Supplementary.

Negative primitives improve accuracy, as indicated by Fig. 7b. This figure shows the histogram of the number of times a particular (K^{total}, K^{-}) combination was selected. Note that there is a strong tendency to use more primitives ($K^{total} = \mathbf{36}$ is much more popular than others). One would expect this from bias considerations, but we observe that

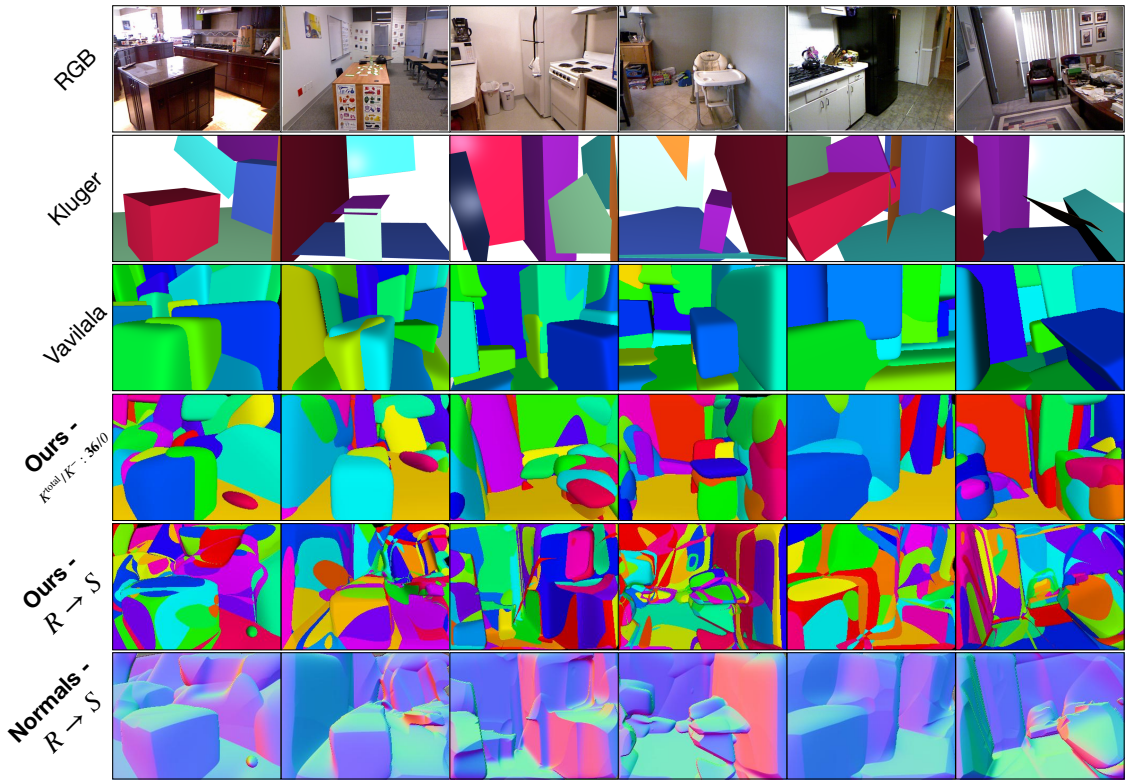


Figure 5. We present a method that advances the SOTA for primitive decomposition by using ensembling and negative primitives. Prior art shown in $\mathbf{2^{nd}}$ row [29] and $\mathbf{3^{rd}}$ row [62] for comparison. (**4th row**) We show results from our approach with **36** primitives, none negative. Our procedure encodes geometry quite closely. (**5th and 6th row**) With negative primitives, we can encode a rich arrangement of shapes. Here, we ensemble many predictors and show the best one. In these six cases, a model with negative primitives was chosen. The normals make it clear that negative primitives are scooping geometry away from positive primitives.

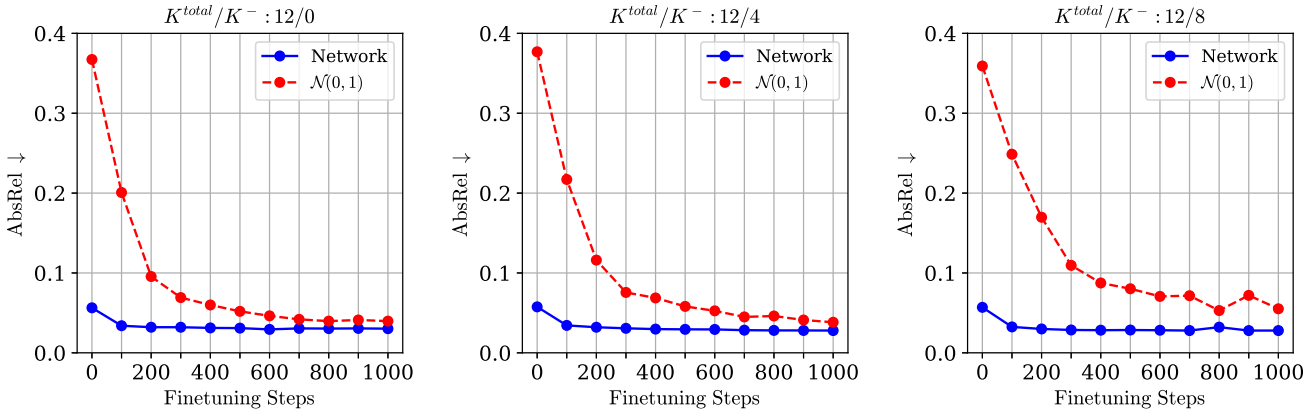


Figure 6. **Network start is beneficial.** Initializing our finetuning process with primitives predicted by our network (blue line) yields better primitives after finetuning than random start (red line). In practice, network start saves around 900 FT steps and can achieve a better quality than random start. It also appears to be harder to fit negative primitives than positives (there is a larger gap in the final AbsRel between the two curves when there are negative primitives). Based on these results, we use 200 finetuning steps to balance compute and quality when reporting error metrics in this paper. Further, the fact that it is even possible to generate high-quality primitives via pure optimization, without a neural network, is new in the context of recent primitive-generation literature. Results shown on 100 random test images from LAION.

Method	K^{total}	K^-	AbsRel↓	Normals Mean↓	Normals Median↓	SegAcc↑	Time (s)	Mem (GB)
12	**12**	*0*	0.0622	*34.4*	26.3	0.651	0.84	3.53
24	**24**	*0*	0.052	33	25	0.7	1.46	5.57
36	**36**	*0*	0.0484	32.3	24.4	0.723	2.06	7.61
best	**36**	*12*	*0.0452*	*32.1*	*24*	**0.765**	2.06	7.61
pos $S \to R$	**26.1**	*0*	0.0527	33	24.9	0.7	2.08	7.61
pos + neg $S \to R$	**29**	*10.4*	0.0492	32.9	24.7	0.733	2.13	7.61
pos $R \to S$	**33.6**	*0*	0.0476	32.3	24.4	0.72	4.37	7.61
pos + neg $R \to S$	**35**	*14.7*	**0.0417**	**31.9**	**24**	0.756	29.9	7.61
Vavilala & Forsyth [62]	**13.9**	*0*	0.098	37.4	32.4	0.618	40	6.77

Table 1. Comparison to SOTA (last row) on NYUv2. **All rows except the last were trained using our procedure.** Our best approach (second last row) polishes then chooses from 18 different models with different numbers of primitives. **First three rows**: we train a primitive generation model according to the procedure laid out in Sec 3, without negative primitives. Next row: **36** total primitives with *12* negative was our best network as measured by AbsRel. **Next four rows** Ensembling strongly improves error metrics, particularly AbsRel. pos + neg refers to all 18 models available for ensembling, whereas pos refers to only 3 models without negative primitives available. $S \to R$ refers to only refining the output of the model with the best sample classification; $R \to S$ means we finetune all models and pick the best one. In this table, we finetune assuming GT depth is available at test time, though our method still works even when depth is inferred by a pretrained depth estimator. The fact that substantial gains can be achieved from $R \to S$ implies that the best start point may not yield the best end point – meaning the fitting problem is hard. Time and memory estimates are presented as well. **Last row**: we compare our methods against existing work. Any individual model we train obtains better error metrics with less compute. Timings for ensembling show estimated total cost of running all the methods and selecting the best one; memory refers to peak GPU memory usage.

Dataset	faces	AbsRel↓
NYUv2	6	0.0417
LAION	6	0.0193
LAION	12	0.0178

Table 2. Depth metrics for NYUv2 data and LAION data compared, when ensembling pos + neg $R \to S$. The much larger data volume in LAION improves the generalization ability of our procedure. Further, increasing the representational power of our method by removing the symmetric normal constraint (for parallelepipeds, top two rows) and increasing the number of faces per polytope to $f = 12$ (bottom row) yields even better primitive decompositions.

K^{total}	Encode	Loss	Finetune	Render
12	0.0006	0.0006	0.68	0.15
24	0.0006	0.0006	1.23	0.22
36	0.0006	0.0007	1.79	0.26

Table 3. Estimated inference breakdown times, all times in seconds, 256-res images, $K^- = 0$. Encoding is very fast, in which the network predicts parameters of the primitives given an RGBD image. Computing loss, required for getting the fraction of samples classified correctly when ensembling with $S \to R$, is also fast. However, finetuning (we show 200 steps here) is often the bottleneck since we must compute the loss and optimize the parameters of the primitives.

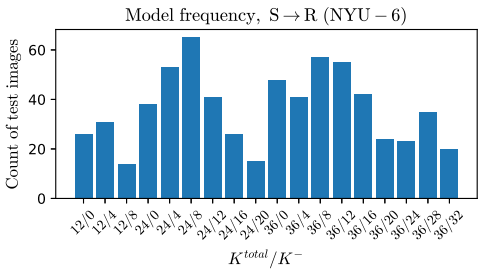
many scenes use fewer primitives. The number of negatives used for the best fit is quite variable, though often near $K^- \approx \frac{1}{3}K^{total}$ - see Sec. 8. We believe the ensembling step reduces variance by discarding complex fits that, while more flexible and thus lower in bias, tend to overfit and become less accurate. When complex fits perform poorly, the ensemble instead favors simpler, more robust models.

Our method is efficient, as Table 3 shows. The vast majority of time is spent in polishing the representation. Ensembling yields further improvements (particularly $S \to R$), and is still more efficient than prior work (see table 1).

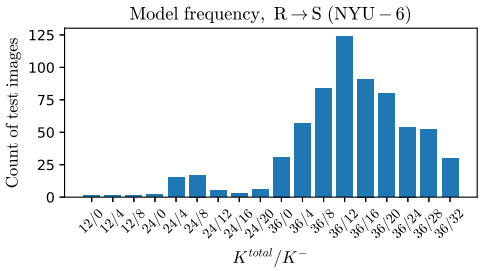
4.3. LAION Results

Scaling is an important concept in computer vision, but we have not seen this concept applied to 3D primitive generation. To that end, we collect approx. 1.8M natural images from LAION-Aesthetic. We use a recent SOTA depth estimation network [70] to obtain depth maps, and make reasonable camera calibration assumptions to lift a 3D point cloud from the depth map. In particular, we use the Hypersim [51] module that predicts metric depth and use its camera parameters to get the point cloud for each image, which is required for training our convex decomposition model. GT normals can be obtained using the image gradient method described in [62], which requires point cloud input. **LAION is easier than NYUv2** as Table 2 shows (note the improved AbsRel), likely due to the much larger dataset and availability of easier scenes. NYUv2 scenes are complex cluttered indoor rooms.

Our depth AbsRel numbers on LAION (comparing primitive depth against supplied depth map) lie between .019 to .029 (see Table 8). For reference, the state-of-the-art depth



(a) Select then refine ensembling on NYUv2.



(b) Refine then select ensembling on NYUv2

Figure 7. Negative primitives are often selected from the ensemble. **Top** When we ensemble with $S \rightarrow R$, all models across all primitive counts are well-represented. This indicates that our network prediction may slightly struggle to manage larger numbers of primitives, hence the relative success of fewer primitives. In this setting, selecting a prediction for finetuning is based on fraction of 3D samples classified incorrectly, which is fast as we don't need to finetune and render the outputs of all the networks to decide which model to use. **Bottom** When we refine then choose $R \rightarrow S$, our finetuning procedure polishes each network start point and chooses the best one based on AbsRel, requiring a render for each model. When doing so, the best model (measured by AbsRel of rendered depth against GT depth) is strongly concentrated among higher primitive counts, $K^{total} = \mathbf{36}$.

estimator DepthAnythingV2 [70] reports AbsRel values (predicted vs. ground truth depth) of approximately 0.044 to 0.075 on NYUv2 and KITTI, which helps contextualize that our depth errors are comparatively small.

Our network start is much better than pure descent, as Fig. 6 shows. The randomly started pure descent procedure of Section 3.2 produces surprisingly strong fits, but requires a large number of iterations to do so. Typically, 100 iterations of polishing a network start point is much better than 1000 iterations of pure descent. The descent procedure is a first order method, so we expect AbsRel to improve no faster than 1/iterations, suggesting that this figure understates the advantage of the network start point.

5. Discussion

Primitives are an old obsession in computer vision. Their original purpose (object recognition) now appears to be much better handled in other ways. Mostly, using primitives was never really an issue, because there weren't viable fitting procedures. But what are primitives for? Likely answers come from robotics – where one might benefit from simplified representations of geometry that are still accurate – and image editing – where a user might edit a scene by moving primitives (e.g. Fig. 8). Future work may better handle variable primitive representations (we train a separate network for each primitive count K^{total}/K^{-}) and improve the network start, eliminating the need for post-training finetuning. While this paper has drastically advanced geometric accuracy of 3D primitive fitting to data, improving segmentation boundaries remains open.

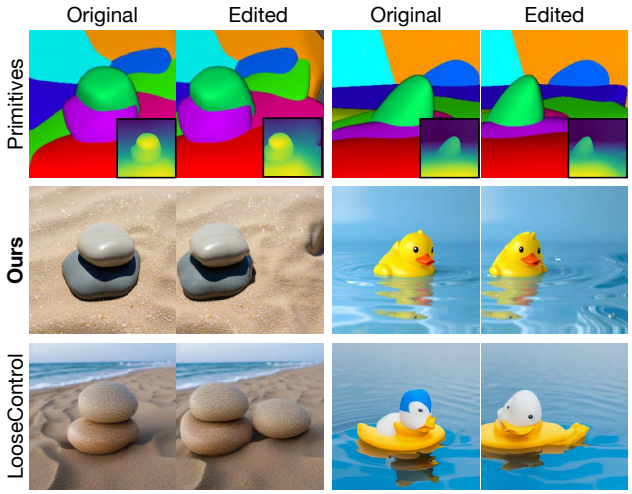


Figure 8. Primitives support image editing. Figure reproduced from concurrent submission [63]. Primitives yield camera moves by a pipeline that: fits primitives to a single RGBD image from a scene; uses those primitives to transfer image texture to an image from the new viewpoint, recording where textures are uncertain or unknown; then synthesizes an image using a depth conditioned diffusion model with the transferred image as a hint. Examples where objects are moved appear in supplementary [63]. This figure compares our primitives with results from LooseControl [3]. Figures show scenes, primitives and depth maps, where "Edited" is post camera move. In each case, the camera is moved to the right. LooseControl is at a disadvantage in this task, because its primitives, while compact, are intentionally not accurate, so the duck changes and an extra rock appears. In each case, our primitive representation is accurate enough to show the same scene from a different view.

6. Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 2106825 and by a gift from the Boeing Corporation. This research used the Delta advanced computing and data resource which is supported by the National Science Foundation (award OAC 2005572) and the State of Illinois.

References

[1] Stephan Alaniz, Massimiliano Mancini, and Zeynep Akata. Iterative superquadric recomposition of 3d objects from multiple views. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 18013–18023, 2023. 3

[2] Barr. Superquadrics and angle-preserving transformations. *IEEE Computer Graphics and Applications*, 1:11–23, 1981. 2

[3] Shariq Farooq Bhat, Niloy J. Mitra, and Peter Wonka. Loosecontrol: Lifting controlnet for generalized depth conditioning, 2023. 1, 4, 8

[4] I Biederman. Recognition by components : A theory of human image understanding. *Psychological Review*, (94): 115–147, 1987. 2

[5] TO Binford. Visual perception by computer. In *IEEE Conf. on Systems and Controls*, 1971. 2

[6] Stéphane Calderon and Tamy Boubekeur. Bounding proxies for shape approximation. *ACM Transactions on Graphics (TOG)*, 36:1 – 13, 2017. 2

[7] Zhiqin Chen, Andrea Tagliasacchi, and Hao Zhang. Bsp-net: Generating compact meshes via binary space partitioning. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 42–51, 2019. 2

[8] Boyang Deng, Simon Kornblith, and Geoffrey Hinton. Cerberus: A multi-headed derenderer. In *Workshop on 3D Scene Understanding*, 2019. 2

[9] Boyang Deng, Kyle Genova, Soroosh Yazdani, Sofien Bouaziz, Geoffrey Hinton, and Andrea Tagliasacchi. Cvxnet: Learnable convex decomposition. 2020. 1, 2, 3

[10] M. A. Fischler and R. C. Bolles. Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Comm. ACM.*, 24(6): 381–395, 1981. 3

[11] David F. Fouhey, Abhinav Gupta, and Martial Hebert. Data-driven 3D primitives for single image understanding. In *ICCV*, 2013. 2

[12] K. Fu, J. Peng, and Q. He et al. Single image 3d object reconstruction based on deep learning: A review. *Multimed Tools Appl*, 80:463–498, 2021. 2

[13] Matheus Gadelha, Giorgio Gori, Duygu Ceylan, Radomír Mech, Nathan A. Carr, Tamy Boubekeur, Rui Wang, and Subhransu Maji. Learning generative models of shape handles. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 399–408, 2020. 2

[14] M. Ghosh, S. Thomas, and N. M. Amato. Fast collision detection for motion planning using shape primitive skeletons. In *International Workshop on the Algorithmic Foundations of Robotics*, pages 36–51. Springer International Publishing, 2018. 1

[15] Abhinav Gupta, Alexei A. Efros, and Martial Hebert. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*, 2010. 2

[16] Shreyas Hampali, Sinisa Stekovic, Sayan Deb Sarkar, Chetan Srinivasa Kumar, Friedrich Fraundorfer, and Vincent Lepetit. Monte carlo scene search for 3d scene understanding. *2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13799–13808, 2021. 3

[17] V. Hedau, D. Hoiem, and D. Forsyth. Recovering the Spatial Layout of Cluttered Rooms. In *Proc. ICCV*, 2009. 2

[18] Varsha Hedau, Derek Hoiem, and David Forsyth. Thinking Inside the Box: Using Appearance Models and Context Based on Room Geometry. In *Proc. ECCV*, 2010. 2

[19] V. Hedau, D. Hoiem, and D. Forsyth. Recovering Free Space of Indoor Scenes from a Single Image. In *Proc. CVPR*, 2012. 2

[20] A. Hertz, O. Perel, O. Sorkine-Hornung, and D. Cohen-Or. Spaghetti: editing implicit shapes through part aware generation. *ACM Transactions on Graphics*, 41(4):1–20, 2022. 2

[21] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Automatic photo pop-up. *ACM Transactions on Graphics / SIGGRAPH*, 24(3), 2005. 2

[22] D. Hoiem, A. A. Efros, and M. Hebert. Recovering surface layout from an image. *IJCV*, 2007. 2

[23] Aleş Jaklič, Aleš Leonardis, and Franc Solina. Segmentation and recovery of superquadrics. In *Computational Imaging and Vision*, 2000. 2

[24] Eric Jang, Shixiang Gu, and Ben Poole. Categorical reparameterization with gumbel-softmax. In *International Conference on Learning Representations*, 2017. 2

[25] Hao Jiang. Finding approximate convex shapes in rgbd images. In *European Conference on Computer Vision*, pages 582–596. Springer, 2014. 3

[26] Zhizhong Kang, Juntao Yang, Zhou Yang, and Sai Cheng. A review of techniques for 3d reconstruction of indoor environments. *ISPRS Int. J. Geo Inf.*, 9:330, 2020. 3

[27] Florian Kluger and Bodo Rosenhahn. PARSAC: Accelerating Robust Multi-Model Fitting with Parallel Sample Consensus. In *AAAI*, 2024. 3

[28] Florian Kluger, Eric Brachmann, Hanno Ackermann, Carsten Rother, Michael Ying Yang, and Bodo Rosenhahn. CONSAC: Robust Multi-Model Fitting by Conditional Sample Consensus. In *CVPR*, 2020.

[29] Florian Kluger, Hanno Ackermann, Eric Brachmann, Michael Ying Yang, and Bodo Rosenhahn. Cuboids revisited: Learning robust 3d shape fitting to single rgb images. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021. 2, 5, 6, 13

[30] Florian Kluger, Eric Brachmann, Michael Ying Yang, and Bodo Rosenhahn. Robust shape fitting for 3d scene abstraction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024. 3

[31] Lingxiao Li, Minhyuk Sung, Anastasia Dubrovina, L. Yi, and Leonidas J. Guibas. Supervised fitting of geometric primitives to 3d point clouds. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2647–2655, 2018. 2

[32] Chen Liu, Kihwan Kim, Jinwei Gu, Yasutaka Furukawa, and Jan Kautz. Planercnn: 3d plane detection and reconstruction from a single image. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4445–4454, 2018. 2

[33] Chen Liu, Jimei Yang, Duygu Ceylan, Ersin Yumer, and Yasutaka Furukawa. Planenet: Piece-wise planar reconstruction from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2579–2588, 2018. 2

[34] Haolin Liu, Yujian Zheng, Guanying Chen, Shuguang Cui, and Xiaoguang Han. Towards high-fidelity single-view holistic reconstruction of indoor scenes. In *European Conference on Computer Vision*, pages 429–446. Springer, 2022. 2

[35] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. 3

[36] D. Marr and H. K. Nishihara. Representation and recognition of the spatial organization of three-dimensional shapes. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 200(1140):269–294, 1978. 4

[37] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen. Automatic grasp planning using shape primitives. In *IEEE International Conference on Robotics and Automation*, 2003. 1

[38] Kaichun Mo, Paul Guerrero, L. Yi, Hao Su, Peter Wonka, Niloy Jyoti Mitra, and Leonidas J. Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *ACM Trans. Graph.*, 38:242:1–242:19, 2019. 2

[39] Tom Monnier, Jake Austin, Angjoo Kanazawa, Alexei Efros, and Mathieu Aubry. Differentiable blocks world: Qualitative 3d decomposition by rendering primitives. *Advances in Neural Information Processing Systems*, 36:5791–5807, 2023. 3

[40] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 5

[41] R. Nevatia and T.O. Binford. Description and recognition of complex curved objects. *Artificial Intelligence*, 1977. 2

[42] Matthias Nieuwenhuisen, Jörg Stückler, Alexander Berner, Reinhard Klein, and Sven Behnke. Shape-primitive based object recognition and grasping. In *Proceedings of the 7th German Conference on Robotics (ROBOTIK)*, pages 1–5, Munich, Germany, 2012. VDE Verlag. 1

[43] Despoina Paschalidou, Ali O. Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 10336–10345, 2019. 2

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

[45] J. Ponce and M. Hebert. A new method for segmenting 3-d scenes into primitives. In *Proc. 6 ICPR*, 1982. 2

[46] Michaël Ramamonjisoa, Sinisa Stekovic, and Vincent Lepetit. Monteboxfinder: Detecting and filtering primitives to fit a noisy point cloud. In *Computer Vision – ECCV 2022*, pages 160–176. Springer, 2022. 3

[47] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020. 5

[48] René Ranftl, Katrin Lasinger, David Hafner, Konrad Schindler, and Vladlen Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(3), 2022. 3

[49] Dominic Roberts, Aram Danielyan, Hang Chu, Mani Golparvar Fard, and David A. Forsyth. Lsd-structurenet: Modeling levels of structural detail in 3d part hierarchies. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 5816–5825, 2021. 2

[50] L. G. Roberts. *Machine Perception of Three-Dimensional Solids*. PhD thesis, MIT, 1963. 2

[51] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Angel Bautista, Nathan Paczan, Russ Webb, and Joshua M. Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding, 2021. 7

[52] S. Shafer and T. Kanade. The theory of straight homogeneous generalized cylinders. In *Technical Report CS-083-105, Carnegie Mellon University*, 1983. 2

[53] Dmitriy Smirnov, Matthew Fisher, Vladimir G. Kim, Richard Zhang, and Justin M. Solomon. Deep parametric shape predictions using distance fields. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 558–567, 2019. 2

[54] Sinisa Stekovic, Shreyas Hampali, Mahdi Rad, Sayan Deb Sarkar, Friedrich Fraundorfer, and Vincent Lepetit. General 3d room layout from a single view by render-and-compare. In *European Conference on Computer Vision*, pages 187–203. Springer, 2020. 2

[55] Chun-Yu Sun and Qian-Fang Zou. Learning adaptive hierarchical cuboid abstractions of 3d shape collections. *ACM Transactions on Graphics (TOG)*, 38:1 – 13, 2019. 2

[56] Maxim Tatarchenko, Stephan R. Richter, René Ranftl, Zhuwen Li, Vladlen Koltun, and Thomas Brox. What do single-view 3d reconstruction networks learn? *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3400–3409, 2019. 2

[57] Maged S. Tawfik. An efficient algorithm for csg to b-rep conversion. In *Proceedings of the First ACM Symposium on Solid Modeling Foundations and CAD/CAM Applications*, page 99–108, New York, NY, USA, 1991. Association for Computing Machinery. 1

[58] Kevin Tracy, Taylor A. Howell, and Zachary Manchester. Differentiable collision detection for a set of convex primitives, 2023. 1

[59] Shubham Tulsiani, Hao Su, Leonidas J. Guibas, Alexei A. Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Computer Vision and Pattern Regognition (CVPR)*, 2017. 2

[60] Mikaela Angelina Uy, Yen-Yu Chang, Minhyuk Sung, Purvi Goel, Joseph G Lambourne, Tolga Birdal, and Leonidas J Guibas. Point2cyl: Reverse engineering 3d objects from point clouds to extrusion cylinders. In *CVPR*, 2022. 2

[61] A. van den Hengel, C. Russell, A. Dick, J. Bastian, L. Fleming D. Poo-ley, and L. Agapito. Part-based modelling of compound scenes from images. In *CVPR*, 2015. 2

[62] Vaibhav Vavilala and David Forsyth. Convex decomposition of indoor scenes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9176–9186, 2023. 2, 3, 5, 6, 7, 10, 13

[63] Vaibhav Vavilala, Seemandhar Jain, Rahul Vasanth, D. A. Forsyth, and Anand Bhattad. Generative blocks world: Moving things around in pictures, 2025. 1, 4, 8

[64] Giulia Vezzani, Ugo Pattacini, Giulia Pasquale, and Lorenzo Natale. Improving superquadric modeling and grasping with prior on object shapes. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6875–6882, 2018. 1

[65] Xiaolong Wang, David Fouhey, and Abhinav Gupta. Designing deep networks for surface normal estimation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 539–547, 2015. 5

[66] X. Wei, M. Liu, Z. Ling, and H. Su. Approximate convex decomposition for 3d meshes with collision-aware concavity and tree search. *ACM Transactions on Graphics*, 41(4), 2022. 2

[67] Rundi Wu, Chang Xiao, and Changxi Zheng. Deepcad: A deep generative network for computer-aided design models. In *ICCV*, 2021. 2

[68] Yifeng Xu, Fan Zhu, Ye Li, Sebastian Ren, Xiaonan Huang, and Yuhao Chen. Rgbsqgrasp: Inferring local superquadric primitives from single rgb image for graspability-aware bin picking, 2025. 1

[69] Lihe Yang, Bingyi Kang, Zilong Huang, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything: Unleashing the power of large-scale unlabeled data. In *CVPR*, 2024. 3

[70] Lihe Yang, Bingyi Kang, Zilong Huang, Zhen Zhao, Xiaogang Xu, Jiashi Feng, and Hengshuang Zhao. Depth anything v2. In *Advances in Neural Information Processing Systems*, 2024. 7, 8

[71] Fenggen Yu, Zhiqin Chen, Manyi Li, Aditya Sanghi, Hooman Shayani, Ali Mahdavi-Amiri, and Hao Zhang. Capri-net: Learning compact cad shapes with adaptive primitive assembly. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11768–11778, 2022. 2

[72] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 900–909, 2017. 2

[73] Chuhang Zou, Ersin Yumer, Jimei Yang, Duygu Ceylan, and Derek Hoiem. 3d-prnn: Generating shape primitives with recurrent neural networks. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2017. 2

[74] Chuhang Zou, Alex Colburn, Qi Shan, and Derek Hoiem. Layoutnet: Reconstructing the 3d room layout from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018. 2