Mesh R-CNN

Georgia Gkioxari Jitendra Malik Justin Johnson

Facebook AI Research (FAIR)

Abstract

Rapid advances in 2D perception have led to systems that accurately detect objects in real-world images. However, these systems make predictions in 2D, ignoring the 3D structure of the world. Concurrently, advances in 3D shape prediction have mostly focused on synthetic benchmarks and isolated objects. We unify advances in these two areas. We propose a system that detects objects in real-world images and produces a triangle mesh giving the full 3D shape of each detected object. Our system, called Mesh R-CNN, augments Mask R-CNN with a mesh prediction branch that outputs meshes with varying topological structure by first predicting coarse voxel representations which are converted to meshes and refined with a graph convolution network operating over the mesh's vertices and edges. We validate our mesh prediction branch on ShapeNet, where we outperform prior work on single-image shape prediction. We then deploy our full Mesh R-CNN system on Pix3D, where we jointly detect objects and predict their 3D shapes. Project page: https://gkioxari.github.io/meshrcnn/.

1. Introduction

The last few years have seen rapid advances in 2D object recognition. We can now build systems that accurately recognize objects [19, 29, 54, 60], localize them with 2D bounding boxes [13, 46] or masks [18], and predict 2D keypoint positions [3, 18, 64] in cluttered, real-world images. Despite their impressive performance, these systems ignore one critical fact: that the world and the objects within it are 3D and extend beyond the XY image plane.

At the same time, there have been significant advances in 3D shape understanding with deep networks. A menagerie of network architectures have been developed for different 3D shape representations, such as voxels [5], point-clouds [8], and meshes [68]; each representation carries its own benefits and drawbacks. However, this diverse and creative set of techniques has primarily been developed on synthetic benchmarks such as ShapeNet [4] consisting of rendered objects in isolation, which are dramatically less complex than natural-image benchmarks used for 2D object recognition like ImageNet [51] and COCO [36].

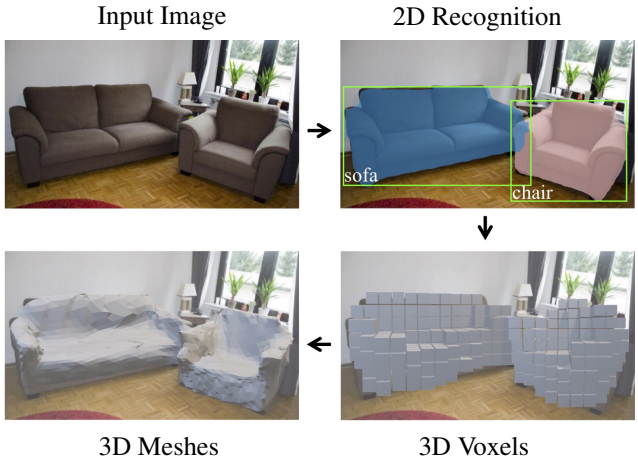

Figure 1. Mesh R-CNN takes an input image, predicts object instances in that image and infers their 3D shape. To capture diversity in geometries and topologies, it first predicts coarse voxels which are refined for accurate mesh predictions.

We believe that the time is ripe for these hitherto distinct research directions to be combined. We should strive to build systems that (like current methods for 2D perception) can operate on unconstrained real-world images with many objects, occlusion, and diverse lighting conditions but that (like current methods for 3D shape prediction) do not ignore the rich 3D structure of the world.

In this paper we take an initial step toward this goal. We draw on state-of-the-art methods for 2D perception and 3D shape prediction to build a system which inputs a real-world RGB image, detects the objects in the image, and outputs a category label, segmentation mask, and a 3D triangle mesh giving the full 3D shape of each detected object.

Our method, called *Mesh R-CNN*, builds on the state-of-the-art Mask R-CNN [18] system for 2D recognition, augmenting it with a *mesh prediction branch* that outputs high-resolution triangle meshes.

Our predicted meshes must be able to capture the 3D structure of diverse, real-world objects. Predicted meshes should therefore dynamically vary their complexity, topology, and geometry in response to varying visual stimuli. However, prior work on mesh prediction with deep networks [23, 56, 68] has been constrained to deform from

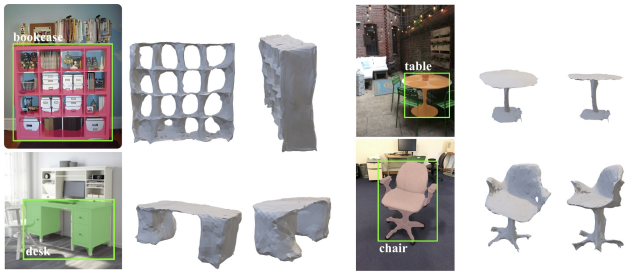

Figure 2. Example predictions from Mesh R-CNN on Pix3D. Using initial voxel predictions allows our outputs to vary in topology; converting these predictions to meshes and refining them allows us to capture fine structures like tabletops and chair legs.

fixed mesh templates, limiting them to fixed mesh topologies. As shown in Figure 1, we overcome this limitation by utilizing multiple 3D shape representations: we first predict coarse *voxelized* object representations, which are converted to meshes and refined to give highly accurate mesh predictions. As shown in Figure 2, this hybrid approach allows Mesh R-CNN to output meshes of arbitrary topology while also capturing fine object structures.

We benchmark our approach on two datasets. First, we evaluate our mesh prediction branch on ShapeNet [4], where our hybrid approach of voxel prediction and mesh refinement outperforms prior work by a large margin. Second, we deploy our full Mesh R-CNN system on the recent Pix3D dataset [59] which aligns 395 models of IKEA furniture to real-world images featuring diverse scenes, clutter, and occlusion. To date Pix3D has primarily been used to evalute shape predictions for models trained on ShapeNet, using perfectly cropped, unoccluded image segments [40, 59, 72], or synthetic rendered images of Pix3D models [75]. In contrast, using Mesh R-CNN we are the first to train a system on Pix3D which can jointly detect objects of all categories and estimate their full 3D shape.

2. Related Work

Our system inputs a single RGB image and outputs a set of detected object instances, with a triangle mesh for each object. Our work is most directly related to recent advances in 2D object recognition and 3D shape prediction. We also draw more broadly from work on other 3D perception tasks.

2D Object Recognition Methods for 2D object recognition vary both in the type of information predicted per object, and in the overall system architecture. Object detectors output per-object bounding boxes and category labels [12, 13, 35, 37, 45, 46]; Mask R-CNN [18] additionally outputs instance segmentation masks. Our method extends this line of work to output a full 3D mesh per object.

Single-View Shape Prediction Recent approaches use a variety of shape representations for single-image 3D reconstruction. Some methods predict the orientation [10, 20] or 3D pose [30, 43, 65] of known shapes. Other approaches predict novel 3D shapes as sets of 3D points [8, 33], patches [15, 69], or geometric primitives [9, 63, 66]; others use deep networks to model signed distance functions [41]. These methods can flexibly represent complex shapes, but rely on post-processing to extract watertight mesh outputs.

Some methods predict regular voxel grids [5, 70, 71]; while intuitive, scaling to high-resolution outputs requires complex octree [49, 61] or nested shape architectures [48].

Others directly output triangle meshes, but are constrained to deform from fixed [55, 56, 68] or retrieved mesh templates [50], limiting the topologies they can represent.

Our approach uses a hybrid of voxel prediction and mesh deformation, enabling high-resolution output shapes that can flexibly represent arbitrary topologies.

Some methods reconstruct 3D shapes without 3D annotations [23, 25, 47, 67, 74]. This is an important direction, but at present we consider only the fully supervised case due to the success of strong supervision for 2D perception.

Multi-View Shape Prediction There is a broad line of work on multi-view reconstruction of objects and scenes, from classical binocular stereo [17, 52] to using shape priors [1, 2, 6, 21] and modern learning techniques [24, 26, 53]. In this work, we focus on single-image shape reconstruction.

3D Inputs Our method inputs 2D images and predicts semantic labels and 3D shapes. Due to the increasing availabilty of depth sensors, there has been growing interest in methods predicting semantic labels from 3D inputs such as RGB-D images [16, 57] and pointclouds [14, 31, 44, 58, 62]. We anticipate that incorporating 3D inputs into our method could improve the fidelity of our shape predictions.

Datasets Advances in 2D perception have been driven by large-scale annotated datasets such as ImageNet [51] and COCO [36]. Datasets for 3D shape prediction have lagged their 2D counterparts due to the difficulty of collecting 3D annotations. ShapeNet [4] is a large-scale dataset of CAD models which are rendered to give synthetic images. The IKEA dataset [32] aligns CAD models of IKEA objects to real-world images; Pix3D [59] extends this idea to a larger set of images and models. Pascal3D [73] aligns CAD models to real-world images, but it is unsuitable for shape reconstruction since its train and test sets share the same small set of models. KITTI [11] annotates outdoor street scenes with 3D bounding boxes, but does not provide shape annotations.

3. Method

Our goal is to design a system that inputs a single image, detects all objects, and outputs a category label, bounding box, segmentation mask and 3D triangle mesh for each detected object. Our system must be able to handle cluttered real-world images, and must be trainable end-to-end. Our output meshes should not be constrained to a fixed topol-

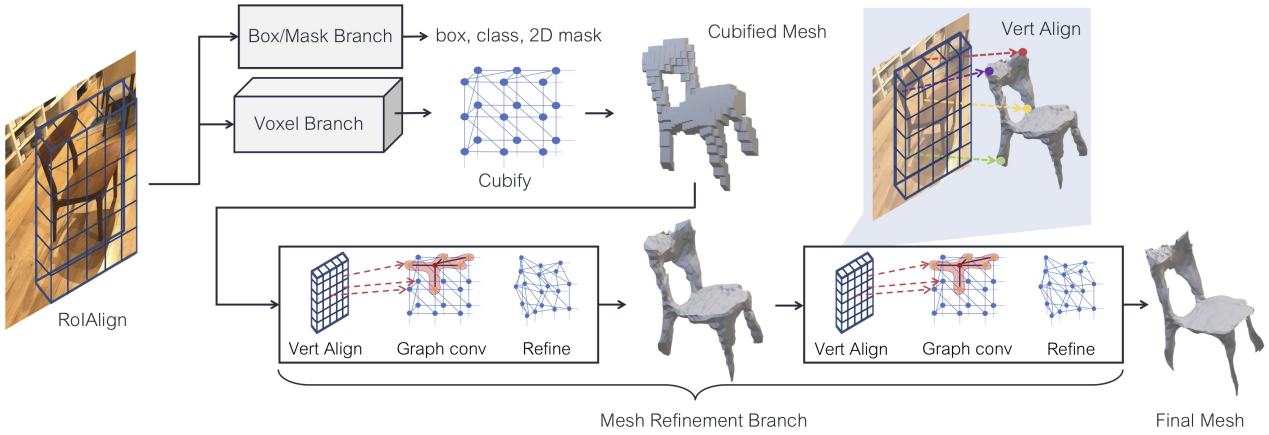

Figure 3. System overview of Mesh R-CNN. We augment Mask R-CNN with 3D shape inference. The *voxel branch* predicts a coarse shape for each detected object which is further deformed with a sequence of refinement stages in the *mesh refinement branch*.

ogy in order to accommodate a wide variety of complex real-world objects. We accomplish these goals by marrying state-of-the-art 2D perception with 3D shape prediction.

Specifically, we build on Mask R-CNN [18], a state-of-the-art 2D perception system. Mask R-CNN is an end-to-end region-based object detector. It inputs a single RGB image and outputs a bounding box, category label, and segmentation mask for each detected object. The image is first passed through a *backbone network* (*e.g.* ResNet-50-FPN [34]); next a *region proposal network* (RPN) [46] gives object proposals which are processed with object classification and mask prediction branches.

Part of Mask R-CNN's success is due to `RoIAlign` which extracts region features from image features while maintaining alignment between the input image and features used in the final prediction branches. We aim to maintain similar feature alignment when predicting 3D shapes.

We infer 3D shapes with a novel mesh predictor, comprising a *voxel branch* and a *mesh refinement branch*. The voxel branch first estimates a coarse 3D voxelization of an object, which is converted to an initial triangle mesh. The mesh refinement branch then adjusts the vertex positions of this initial mesh using a sequence of graph convolution layers operating over the edges of the mesh.

The voxel branch and mesh refinement branch are homologous to the existing box and mask branches of Mask R-CNN. All take as input image-aligned features corresponding to RPN proposals. The voxel and mesh losses, described in detail below, are added to the box and mask losses and the whole system is trained end-to-end. The output is a set of boxes along with their predicted object scores, masks and 3D shapes. We call our system *Mesh R-CNN*, which is illustrated in Figure 3.

We now describe our mesh predictor, consisting of the voxel branch and mesh refinement branch, along with its associated losses in detail.

3.1. Mesh Predictor

At the core of our system is a mesh predictor which receives convolutional features aligned to an object's bounding box and predicts a triangle mesh giving the object's full 3D shape. Like Mask R-CNN, we maintain correspondence between the input image and features used at all stages of processing via region- and vertex-specific alignment operators (`RoIAlign` and `VertAlign`). Our goal is to capture instance-specific 3D shapes of all objects in an image. Thus, each predicted mesh must have instance-specific *topology* (genus, number of vertices, faces, connected components) and *geometry* (vertex positions).

We predict varying mesh topologies by deploying a sequence of shape inference operations. First, the *voxel branch* makes bottom-up voxelized predictions of each object's shape, similar to Mask R-CNN's mask branch. These predictions are converted into meshes and adjusted by the *mesh refinement head*, giving our final predicted meshes.

The output of the mesh predictor is a *triangle mesh* $T = (V, F)$ for each object. $V = \{v_i \in \mathbb{R}^3\}$ is the set of vertex positions and $F \subseteq V \times V \times V$ is a set of triangular faces.

3.1.1 Voxel Branch

The *voxel branch* predicts a grid of voxel occupancy probabilities giving the course 3D shape of each detected object. It can be seen as a 3D analogue of Mask R-CNN's mask prediction branch: rather than predicting a $M \times M$ grid giving the object's shape in the image plane, we instead predict a $G \times G \times G$ grid giving the object's full 3D shape.

Like Mask R-CNN, we maintain correspondence between input features and predicted voxels by applying a small fully-convolutional network [38] to the input feature map resulting from `RoIAlign`. This network produces a feature map with G channels giving a column of voxel occupancy scores for each position in the input.

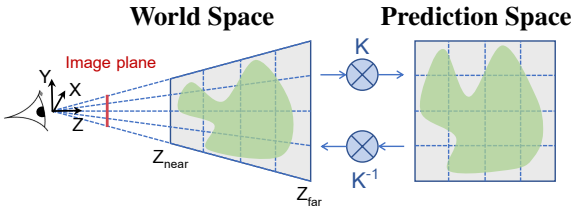

Figure 4. Predicting voxel occupancies aligned to the image plane requires an irregularly-shaped voxel grid. We achieve this effect by making voxel predictions in a space that is transformed by the camera's (known) intrinsic matrix K. Applying K^{-1} transforms our predictions back to world space. This results in frustum-shaped voxels in world space.

Maintaining pixelwise correspondence between the image and our predictions is complex in 3D since objects become smaller as they recede from the camera. As shown in Figure 4, we account for this by using the camera's (known) intrinsic matrix to predict frustum-shaped voxels.

Cubify: Voxel to Mesh The voxel branch produces a 3D grid of occupancy probabilities giving the coarse shape of an object. In order to predict more fine-grained 3D shapes, we wish to convert these voxel predictions into a triangle mesh which can be passed to the mesh refinement branch.

We bridge this gap with an operation called `cubify`. It inputs voxel occupancy probabilities and a threshold for binarizing voxel occupancy. Each occupied voxel is replaced with a cuboid triangle mesh with 8 vertices, 18 edges, and 12 faces. Shared vertices and edges between adjacent occupied voxels are merged, and shared interior faces are eliminated. This results in a watertight mesh whose topology depends on the voxel predictions.

`Cubify` must be efficient and batched. This is not trivial and we provide technical implementation details of how we achieve this in Appendix. Alternatively marching cubes [39] could extract an isosurface from the voxel grid, but is significantly more complex.

Voxel Loss The voxel branch is trained to minimize the binary cross-entropy between predicted voxel occupancy probabilities and true voxel occupancies.

3.1.2 Mesh Refinement Branch

The cubified mesh from the voxel branch only provides a coarse 3D shape, and it cannot accurately model fine structures like chair legs. The *mesh refinement branch* processes this initial cubified mesh, refining its vertex positions with a sequence of refinement stages. Similar to [68], each refinement stage consists of three operations: *vertex alignment*, which extracts image features for vertices; *graph convolution*, which propagates information along mesh edges; and *vertex refinement*, which updates vertex positions. Each layer of the network maintains a 3D position v_i and a feature vector f_i for each mesh vertex.

Vertex Alignment yields an image-aligned feature vector for each mesh vertex[1]. We use the camera's intrinsic matrix to project each vertex onto the image plane. Given a feature map, we compute a bilinearly interpolated image feature at each projected vertex position [22].

In the first stage of the mesh refinement branch, `VertAlign` outputs an initial feature vector for each vertex. In subsequent stages, the `VertAlign` output is concatenated with the vertex feature from the previous stage.

Graph Convolution [28] propagates information along mesh edges. Given input vertex features $\{f_i\}$ it computes updated features $f_i' = \text{ReLU}\left(W_0 f_i + \sum_{j \in \mathcal{N}(i)} W_1 f_j\right)$ where $\mathcal{N}(i)$ gives the i-th vertex's neighbors in the mesh, and W_0 and W_1 are learned weight matrices. Each stage of the mesh refinement branch uses several graph convolution layers to aggregate information over local mesh regions.

Vertex Refinement computes updated vertex positions $v_i' = v_i + \tanh(W_{vert}\,[f_i; v_i])$ where W_{vert} is a learned weight matrix. This updates the mesh *geometry*, keeping its *topology* fixed. Each stage of the mesh refinement branch terminates with vertex refinement, producing an intermediate mesh output which is further refined by the next stage.

Mesh Losses Defining losses that operate natively on triangle meshes is challenging, so we instead use loss functions defined over a finite set of points. We represent a mesh with a pointcloud by densely sampling its surface. Consequently, a pointcloud loss approximates a loss over shapes.

Similar to [56], we use a differentiable *mesh sampling* operation to sample points (and their normal vectors) uniformly from the surface of a mesh. To this end, we implement an efficient batched sampler; see Appendix for details. We use this operation to sample a pointcloud P^{gt} from the ground-truth mesh, and a pointcloud P^i from each intermediate mesh prediction from our model.

Given two pointclouds P, Q with normal vectors, let $\Lambda_{P,Q} = \{(p, \arg\min_q \|p - q\|) : p \in P\}$ be the set of pairs (p, q) such that q is the nearest neighbor of p in Q, and let u_p be the unit normal to point p. The *chamfer distance* between pointclouds P and Q is given by

$$\mathcal{L}_{\text{cham}}(P, Q) = |P|^{-1} \sum_{(p,q)\in\Lambda_{P,Q}} \|p - q\|^2 + |Q|^{-1} \sum_{(q,p)\in\Lambda_{Q,P}} \|q - p\|^2 \quad (1)$$

and the (absolute) *normal distance* is given by

$$\mathcal{L}_{\text{norm}}(P, Q) = -|P|^{-1} \sum_{(p,q)\in\Lambda_{P,Q}} |u_p \cdot u_q| - |Q|^{-1} \sum_{(q,p)\in\Lambda_{Q,P}} |u_q \cdot u_p|. \quad (2)$$

The chamfer and normal distances penalize mismatched positions and normals between two pointclouds, but minimizing these distances alone results in degenerate meshes (see Figure 5). High-quality mesh predictions require additional *shape regularizers*: To this end we use an *edge loss* $\mathcal{L}_{\text{edge}}(V, E) = \frac{1}{|E|} \sum_{(v,v')\in E} \|v - v'\|^2$ where $E \subseteq V \times V$

[1] Vertex alignment is called *perceptual feature pooling* in [68]

are the edges of the predicted mesh. Alternatively, a Laplacian loss [7] also imposes smoothness constraints.

The *mesh loss* of the i-th stage is a weighted sum of $\mathcal{L}_{\text{cham}}(P^i, P^{gt})$, $\mathcal{L}_{\text{norm}}(P^i, P^{gt})$ and $\mathcal{L}_{\text{edge}}(V^i, E^i)$. The mesh refinement branch is trained to minimize the mean of these losses across all refinement stages.

4. Experiments

We benchmark our mesh predictor on ShapeNet [4], where we compare with state-of-the-art approaches. We then evaluate our full Mesh R-CNN for the task of 3D shape prediction *in the wild* on the challenging Pix3D dataset [59].

4.1. ShapeNet

ShapeNet [4] provides a collection of 3D shapes, represented as textured CAD models organized into semantic categories following WordNet [42], and has been widely used as a benchmark for 3D shape prediction. We use the subset of ShapeNetCore.v1 and rendered images from [5]. Each mesh is rendered from up to 24 random viewpoints, giving RGB images of size 137×137. We use the train / test splits provided by [68], which allocate 35,011 models (840,189 images) to train and 8,757 models (210,051 images) to test; models used in train and test are disjoint. We reserve 5% of the training models as a validation set.

The task on this dataset is to input a single RGB image of a rendered ShapeNet model on a blank background, and output a 3D mesh for the object in the camera coordinate system. During training the system is supervised with pairs of images and meshes.

Evaluation We adopt evaluation metrics used in recent work [55, 56, 68]. We sample 10k points uniformly at random from the surface of predicted and ground-truth meshes, and use them to compute Chamfer distance (Equation 1), Normal consistency, (one minus Equation 2), and F1^τ at various distance thresholds τ, which is the harmonic mean of the precision at τ (fraction of predicted points within τ of a ground-truth point) and the recall at τ (fraction of ground-truth points within τ of a predicted point). Lower is better for Chamfer distance; higher is better for all other metrics.

With the exception of normal consistency, these metrics depend on the absolute scale of the meshes. In Table 1 we follow [68] and rescale by a factor of 0.57; for all other results we follow [8] and rescale so the longest edge of the ground-truth mesh's bounding box has length 10.

Implementation Details Our backbone feature extractor is ResNet-50 pretrained on ImageNet. Since images depict a single object, the voxel branch receives the entire `conv5_3` feature map, bilinearly resized to 24×24, and predicts a $48 \times 48 \times 48$ voxel grid. The `VertAlign` operator concatenates features from `conv2_3`, `conv3_4`, `conv4_6`, and `conv5_3` before projecting to a vector of dimension 128. The mesh refinement branch has three stages, each with six graph convolution layers (of dimension 128) organized into three residual blocks. We train for 25 epochs using Adam [27] with learning rate 10^{-4} and 32 images per batch on 8 Tesla V100 GPUs. We set the `cubify` threshold to 0.2 and weight the losses with $\lambda_{\text{voxel}} = 1$, $\lambda_{\text{cham}} = 1$, $\lambda_{\text{norm}} = 0$, and $\lambda_{\text{edge}} = 0.5$.

	Chamfer (↓)	F1^τ (↑)	$\text{F1}^{2\tau}$ (↑)
N3MR [25]	2.629	33.80	47.72
3D-R2N2 [5]	1.445	39.01	54.62
PSG [8]	0.593	48.58	69.78
Pixel2Mesh [68]†	0.591	59.72	74.19
MVD [55]	-	66.39	-
GEOMetrics [56]	-	67.37	-
Pixel2Mesh [68]‡	0.463	67.89	79.88
Ours (Best)	**0.306**	**74.84**	**85.75**
Ours (Pretty)	0.391	69.83	81.76

Table 1. Single-image shape reconstruction results on ShapeNet, using the evaluation protocol from [68]. For [68], † are results reported in their paper and ‡ is the model released by the authors.

Baselines We compare with previously published methods for single-image shape prediction. N3MR [25] is a weakly supervised approach that fits a mesh via a differentiable renderer without 3D supervision. 3D-R2N2 [5] and MVD [55] output voxel predictions. PSG [8] predicts point-clouds. Appendix additionally compares with OccNet [41].

Pixel2Mesh [68] predicts meshes by deforming and subdividing an initial ellipsoid. GEOMetrics [56] extends [68] with adaptive face subdivision. Both are trained to minimize Chamfer distances; however [68] computes it using predicted mesh vertices, while [56] uses points sampled uniformly from predicted meshes. We adopt the latter as it better matches test-time evaluation. Unlike ours, these methods can only predict connected meshes of genus zero.

The training recipe and backbone architecture vary among prior work. Therefore for a fair comparison with our method we also compare against several ablated versions of our model (see Appendix for exact details):

- Voxel-Only: A version of our method that terminates with the cubified meshes from the voxel branch.
- Pixel2Mesh$^+$: We reimplement Pixel2Mesh [68]; we outperform their original model due to a deeper backbone, better training recipe, and minimizing Chamfer on sampled rather than vertex positions.
- Sphere-Init: Similar to Pixel2Mesh$^+$, but initializes from a high-resolution sphere mesh, performing three stages of vertex refinement without subdivision.
- Ours (light): Uses a smaller nonresidual mesh refinement branch with three graph convolution layers per stage. We will adopt this lightweight design on Pix3D.

Voxel-Only is essentially a version of our method that omits the mesh refinement branch, while Pixel2Mesh$^+$ and Sphere-Init omit the voxel prediction branch.

		Full Test Set							Holes Test Set						
		Chamfer(↓)	Normal	$F1^{0.1}$	$F1^{0.3}$	$F1^{0.5}$	$\|V\|$	$\|F\|$	Chamfer(↓)	Normal	$F1^{0.1}$	$F1^{0.3}$	$F1^{0.5}$	$\|V\|$	$\|F\|$
	Pixel2Mesh [68]‡	0.205	0.736	33.7	80.9	91.7	$2466_{\pm 0}$	$4928_{\pm 0}$	0.272	0.689	31.5	75.9	87.9	$2466_{\pm 0}$	$4928_{\pm 0}$
	Voxel-Only	0.916	0.595	7.7	33.1	54.9	$1987_{\pm 936}$	$3975_{\pm 1876}$	0.760	0.592	8.2	35.7	59.5	$2433_{\pm 925}$	$4877_{\pm 1856}$
Best	Sphere-Init	**0.132**	0.711	38.3	86.5	**95.1**	$2562_{\pm 0}$	$5120_{\pm 0}$	0.138	0.705	40.0	85.4	94.3	$2562_{\pm 0}$	$5120_{\pm 0}$
Best	Pixel2Mesh$^+$	**0.132**	0.707	38.3	86.6	**95.1**	$2562_{\pm 0}$	$5120_{\pm 0}$	0.137	0.696	39.3	85.5	94.4	$2562_{\pm 0}$	$5120_{\pm 0}$
Best	Ours (light)	0.133	0.725	**39.2**	**86.8**	**95.1**	$1894_{\pm 925}$	$3791_{\pm 1855}$	**0.130**	0.723	41.6	**86.7**	94.8	$2273_{\pm 899}$	$4560_{\pm 1805}$
Best	Ours	0.133	**0.729**	38.8	86.6	**95.1**	$1899_{\pm 928}$	$3800_{\pm 1861}$	**0.130**	**0.725**	**41.7**	**86.7**	**94.9**	$2291_{\pm 903}$	$4595_{\pm 1814}$
Pretty	Sphere-Init	0.175	0.718	34.5	82.2	92.9	$2562_{\pm 0}$	$5120_{\pm 0}$	0.186	0.684	34.4	80.2	91.7	$2562_{\pm 0}$	$5120_{\pm 0}$
Pretty	Pixel2Mesh$^+$	0.175	**0.727**	34.9	82.3	92.9	$2562_{\pm 0}$	$5120_{\pm 0}$	0.196	0.685	34.4	79.9	91.4	$2562_{\pm 0}$	$5120_{\pm 0}$
Pretty	Ours (light)	0.176	0.699	34.8	82.4	93.1	$1891_{\pm 924}$	$3785_{\pm 1853}$	0.178	0.688	36.3	82.0	92.4	$2281_{\pm 895}$	$4576_{\pm 1798}$
Pretty	Ours	**0.171**	0.713	**35.1**	**82.6**	**93.2**	$1896_{\pm 928}$	$3795_{\pm 1861}$	**0.171**	**0.700**	**37.1**	**82.4**	**92.7**	$2292_{\pm 902}$	$4598_{\pm 1812}$

Table 2. We report results both on the full ShapeNet test set (left), as well as a subset of the test set consisting of meshes with visible holes (right). We compare our full model with several ablated version: Voxel-Only omits the mesh refinement head, while Sphere-Init and Pixel2Mesh$^+$ omit the voxel head. We show results both for *Best* models which optimize for metrics, as well as *Pretty* models that strike a balance between shape metrics and mesh quality (see Figure 5); these two categories of models should not be compared. We also report the number of vertices $|V|$ and faces $|F|$ in predicted meshes (mean$_{\pm \text{std}}$). ‡ refers to the released model by the authors.

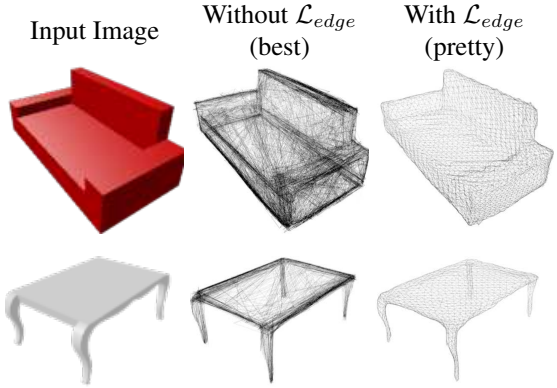

Figure 5. Training without the edge length regularizer $\mathcal{L}_{edge}$ results in degenerate predicted meshes that have many overlapping faces. Adding $\mathcal{L}_{edge}$ eliminates this degeneracy but results in worse agreement with the ground-truth as measured by standard metrics such as Chamfer distance.

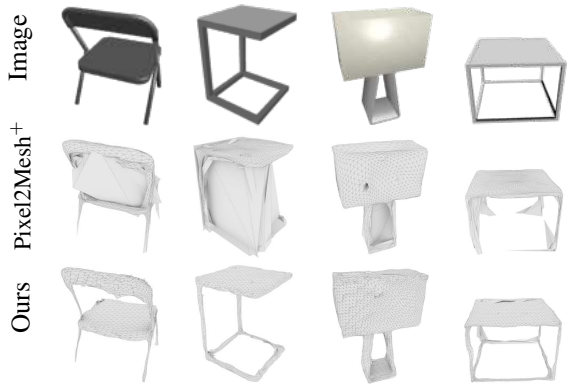

Figure 6. Pixel2Mesh$^+$ predicts meshes by deforming an initial sphere, so it cannot properly model objects with holes. In contrast our method can model objects with arbitrary topologies.

Best vs Pretty As previously noted in [68] (Section 4.1), standard metrics for shape reconstruction are not well-correlated with mesh quality. Figure 5 shows that models trained without shape regularizers give meshes that are preferred by metrics despite being highly degenerate, with irregularly-sized faces and many self-intersections. These degenerate meshes would be difficult to texture, and may not be useful for downstream applications.

Due to the strong effect of shape regularizers on both mesh quality and quantitative metrics, we suggest only quantitatively comparing methods trained with the same shape regularizers. We thus train two versions of all our ShapeNet models: a *Best* version with $\lambda_{\text{edge}} = 0$ to serve as an upper bound on quantitative performance, and a *Pretty* version that strikes a balance between quantitative performance and mesh quality by setting $\lambda_{\text{edge}} = 0.5$.

Comparison with Prior Work Table 1 compares our *Pretty* and *Best* models with prior work on shape prediction from a single image. We use the evaluation protocol from [68], using a 0.57 mesh scaling factor and threshold value $\tau = 10^{-4}$ on squared Euclidean distances. For Pixel2Mesh, we provide the performance reported in their paper [68] as well as the performance of their open-source pretrained model. Table 1 shows that we outperform prior work by a wide margin, validating the design of our mesh predictor.

Ablation Study Fairly comparing with prior work is challenging due to differences in backbone networks, losses, and shape regularizers. For a controlled evaluation, we ablate variants using the same backbone and training recipe, shown in Table 2. ShapeNet is dominated by simple objects of genus zero. Therefore we evaluate both on the entire test set and on a subset consisting of objects with one or more holes (**Holes Test Set**) [2]. In this evaluation we remove the ad-hoc scaling factor of 0.57, and we rescale meshes so the longest edge of the ground-truth mesh's bounding box has length 10, following [8]. We compare the open-source

[2]We annotated 3075 test set models and flagged whether they contained holes. This resulted in 17% (or 534) of the models being flagged. See Appendix for more details and examples.

Pix3D $\mathcal{S}_1$	AP^{box}	AP^{mask}	AP^{mesh}	*chair*	*sofa*	*table*	*bed*	*desk*	*bkcs*	*wrdrb*	*tool*	*misc*	$\lvert V\rvert$	$\lvert F\rvert$
Voxel-Only	93.7	87.1	6.8	0.1	3.6	4.6	3.1	2.0	38.0	7.9	0.0	1.8	$2354_{\pm 706}$	$4717_{\pm 1423}$
Pixel2Mesh$^+$	91.9	86.8	40.4	29.9	63.3	42.9	39.6	33.6	42.2	47.1	36.9	27.7	$2562_{\pm 0}$	$5120_{\pm 0}$
Sphere-Init	92.1	88.2	40.5	33.3	61.9	46.2	40.2	31.0	47.6	34.4	**45.5**	24.0	$2562_{\pm 0}$	$5120_{\pm 0}$
Mesh R-CNN (ours)	92.5	87.5	**55.4**	**48.3**	**76.4**	**68.0**	**51.5**	**47.2**	**71.3**	**60.1**	43.9	**31.7**	$2367_{\pm 698}$	$4743_{\pm 1406}$
# test instances	2440	2440	2440	1129	398	398	205	148	79	53	11	19		
Pix3D $\mathcal{S}_2$														
Voxel-Only	66.4	62.8	4.9	0.0	0.0	2.1	1.4	1.5	18.2	0.2	21.0	0.0	$2346_{\pm 630}$	$4702_{\pm 1269}$
Pixel2Mesh$^+$	67.1	60.8	23.7	22.4	69.4	13.0	42.5	8.6	26.7	1.1	29.6	0.0	$2562_{\pm 0}$	$5120_{\pm 0}$
Sphere-Init	65.9	61.3	24.8	24.6	73.3	13.6	40.2	5.7	31.2	1.5	**33.2**	0.0	$2562_{\pm 0}$	$5120_{\pm 0}$
Mesh R-CNN (ours)	66.4	60.9	**28.7**	**36.6**	**80.1**	**26.5**	**42.8**	**15.6**	**32.4**	**1.8**	22.5	0.0	$2358_{\pm 633}$	$4726_{\pm 1274}$
# test instances	2368	2368	2368	778	506	398	219	205	85	135	22	20		

Table 3. Performance on Pix3D $\mathcal{S}_1$ & $\mathcal{S}_2$. We report mean AP^{box}, AP^{mask} and AP^{mesh}, as well as per category AP^{mesh}. All AP performances are in %. The *Voxel-Only* baseline outputs the cubified voxel predictions. The *Sphere-Init* and *Pixel2Mesh*$^+$ baselines deform an initial sphere and thus are limited to making predictions homeomorphic to spheres. Our Mesh R-CNN is flexible and can capture arbitrary topologies. We outperform the baselines consistently while predicting meshes with fewer number of vertices and faces.

CNN init	# refine steps	AP^{box}	AP^{mask}	AP^{mesh}
COCO	3	92.5	87.5	55.4
IN	3	91.8	85.5	52.9
COCO	2	92.0	86.9	54.5
COCO	1	92.7	87.8	52.4

Table 4. Ablations of Mesh R-CNN on Pix3D.

Pixel2Mesh model against our ablations in this evaluation setting. Pixel2Mesh$^+$ (our reimplementation of [68]) significantly outperforms the original due to an improved training recipe and deeper backbone.

We draw several conclusions from Table 2: (a) On the Full Test Set, our full model and Pixel2Mesh$^+$ perform on par. However, on the Holes Test Set, our model dominates as it is able to predict topologically diverse shapes while Pixel2Mesh$^+$ is restricted to make predictions homeomorphic to spheres, and cannot model holes or disconnected components (see Figure 6). This discrepancy is quantitatively more salient on Pix3D (Section 4.2) as it contains more complex shapes. (b) Sphere-Init and Pixel2Mesh$^+$ perform similarly overall (both *Best* and *Pretty*), suggesting that mesh subdivision may be unnecessary for strong quantitative performance. (c) The deeper residual mesh refinement architecture (inspired by [68]) performs on-par with the lighter non-residual architecture, motivating our use of the latter on Pix3D. (d) Voxel-Only performs poorly compared to methods that predict meshes, demonstrating that mesh predictions better capture fine object structure. (e) Each *Best* model outperforms its corresponding *Pretty* model; this is expected since *Best* is an upper bound on quantitative performance.

4.2. Pix3D

We now turn to Pix3D [59], which consists of 10069 real-world images and 395 unique 3D models. Here the task is to jointly detect and predict 3D shapes for known object categories. Pix3D does not provide standard train/test splits, so we prepare two splits of our own.

Our first split, $\mathcal{S}_1$, randomly allocates 7500 images for training and 2500 for testing. Despite the small number of unique object models compared to ShapeNet, $\mathcal{S}_1$ is challenging since the same model can appear with varying appearance (*e.g.* color, texture), in different orientations, under different lighting conditions, in different contexts, and with varying occlusion. This is a stark contrast with ShapeNet, where objects appear against blank backgrounds.

Our second split, $\mathcal{S}_2$, is even more challenging: we ensure that the 3D models appearing in the train and test sets are disjoint. Success on this split requires generalization not only to the variations present in $\mathcal{S}_1$, but also to novel 3D shapes of known categories: for example a model may see kitchen chairs during training but must recognize armchairs during testing. This split is possible due to Pix3D's unique annotation structure, and poses interesting challenges for both 2D recognition and 3D shape prediction.

Evaluation We adopt metrics inspired by those used for 2D recognition: AP^{box}, AP^{mask} and AP^{mesh}. The first two are standard metrics used for evaluating COCO object detection and instance segmentation at intersection-over-union (IoU) 0.5. AP^{mesh} evalutes 3D shape prediction: it is the mean area under the per-category precision-recall curves for $F1^{0.3}$ at 0.5[3]. Pix3D is not exhaustively annotated, so for evaluation we only consider predictions with box IoU > 0.3 with a ground-truth region. This avoids penalizing the model for correct predictions corresponding to unannotated objects.

We compare predicted and ground-truth meshes in the *camera coordinate system*. Our model assumes known camera intrinsics for `VertAlign`. Mesh R-CNN predicts object positions in the image plane, but it cannot resolve the fundamental scale / depth ambiguity along the Z-axis. During evaluation we therefore match the depth extent (Z_{near} and Z_{far}) of our predictions to the ground-truth shape. Future work might predict depth extents based on shape priors.

[3] A mesh prediction is considered a true-positive if its predicted label is correct, it is not a duplicate detection, and its $F1^{0.3} > 0.5$

Figure 7. Examples of Mesh R-CNN predictions on Pix3D. Mesh R-CNN detects multiple objects per image, reconstructs fine details such as chair legs, and predicts varying and complex mesh topologies for objects with holes such as bookcases and tables.

Pix3D $\mathcal{S}_1$ gt	AP^{box}	AP^{mask}	AP^{mesh}	*chair*	*sofa*	*table*	*bed*	*desk*	*bkcs*	*wrdrb*	*tool*	*misc*	Chamfer (↓)	Normal	$F1^{0.1}$	$F1^{0.3}$	$F1^{0.5}$
Voxel-Only	100.0	90.7	6.7	0.0	2.8	3.9	1.1	1.1	36.7	12.1	2.3	0.6	1.28	0.57	9.9	37.3	56.1
Pixel2Mesh$^+$	100.0	92.0	35.1	22.4	55.6	42.2	32.6	32.5	44.6	38.6	29.1	18.4	1.30	0.70	16.4	51.0	68.4
Sphere-Init	100.0	92.4	33.4	23.7	52.0	41.6	34.9	26.4	42.0	32.9	**33.2**	13.8	1.30	0.69	16.8	51.4	68.8
Mesh R-CNN (ours)	100.0	92.1	**49.1**	**38.8**	**67.0**	**63.4**	**38.9**	**47.2**	**78.3**	**53.7**	**33.2**	**21.1**	**1.11**	**0.71**	**18.7**	**56.4**	**73.5**

Table 5. Performance on Pix3D $\mathcal{S}_1$ on ground-truth regions. In addition to the mean AP^{box}, AP^{mask}, AP^{mesh}, and per category AP^{mesh} we report Chamfer distance, Normal consistency and F1 scores.

Implementation details We use ResNet-50-FPN [34] as the backbone CNN; the box and mask branches are identical to Mask R-CNN. The voxel branch resembles the mask branch, but the pooling resolution is decreased to 12 (*vs*. 14 for masks) due to memory constraints giving $24 \times 24 \times 24$ voxel predictions. We adopt the lightweight design for the mesh refinement branch from Section 4.1. We train for 12 epochs with a batch size of 64 per image on 8 Tesla V100 GPUs (two images per GPU). We use SGD with momentum, linearly increasing the learning rate from 0.002 to 0.02 over the first 1K iterations, then decaying by a factor of 10 at 8K and 10K iterations. We initialize from a model pretrained for instance segmentation on COCO. We set the `cubify` threshold to 0.2 and the loss weights to $\lambda_{\text{voxel}} = 3.0$, $\lambda_{\text{cham}} = 1.0$, $\lambda_{\text{norm}} = 0.1$ and $\lambda_{\text{edge}} = 0.5$ and use weight decay 10^{-4}; detection loss weights are identical to Mask R-CNN.

Comparison to Baselines As discussed in Section 1, we are the first to tackle joint detection and shape inference *in the wild* on Pix3D. To validate our approach we compare with ablated versions of Mesh R-CNN, replacing our full mesh predictor with *Voxel-Only*, *Pixel2Mesh*$^+$, and *Sphere-Init* branches (see Section 4.1). All baselines otherwise use the same architecture and training recipe.

Table 3 (top) shows the performance on $\mathcal{S}_1$. We observe that: (a) Mesh R-CNN outperforms all baselines, improving over the next-best by 14.9% AP^{mesh} overall and across most categories; *Tool* and *Misc*[4] have very few test-set instances (11 and 19 respectively), so their AP is noisy. (b) Mesh R-CNN shows large gains *vs*. *Sphere-Init* for objects with complex shapes such as *bookcase* (+23.7%), *table* (+21.8%) and *chair* (+15.0%). (c) Voxel-Only performs very poorly – this is expected due to its coarse predictions.

Table 3 (bottom) shows the performance on the more challenging $\mathcal{S}_2$ split. Here we observe: (a) The overall performance on 2D recognition (AP^{box} , AP^{mask}) drops significantly compared to $\mathcal{S}_1$, signifying the difficulty of recognizing novel shapes in the wild. (b) Mesh R-CNN outperforms all baselines for shape prediction for all categories except *tool*. (c) Absolute performance on *wardrobe* and *misc* is small for all methods due to significant shape disparity between models in train and test.

Table 4 compares pretraining on COCO vs ImageNet, and compares different architectures for the mesh predictor. COCO vs ImageNet initialization significantly improves 2D recognition (AP^{mask} 87.5 *vs*. 85.5) and 3D shape prediction (AP^{mesh} 55.4 *vs*. 52.9). Shape prediction is significantly degraded when using only one mesh refinement stage (AP^{mesh} 55.4 *vs*. 52.4).

In Table 5 we evaluate our trained models using ground-truth object regions, thus assuming perfect bounding-box detection. Absolute performance on shape reconstruction (Chamfer, Normal, *etc*.) is significantly lower than on ShapeNet, demonstrating the difficulty of Pix3D. AP^{mesh} drops by a few points for all models compared to Table 3 (top), possibly because tight object regions, devoid of context, are not ideal for 3D shape prediction, which can be amplified when training on imperfect region proposals.

Figures 2 and 7 show example predictions from Mesh R-CNN. Our method can detect multiple objects per image, reconstruct fine details such as chair legs, and predict varying and complex mesh topologies for objects with holes such as bookcases and desks.

Discussion

We propose Mesh R-CNN, a novel system for joint 2D perception and 3D shape inference. We validate our approach on ShapeNet and show its merits on Pix3D. Mesh R-CNN is a first attempt at 3D shape prediction in the wild. Despite the lack of large supervised data, *e.g*. compared to COCO, Mesh R-CNN shows promising results.

[4] *Misc* consists of objects such as fire hydrant, picture frame, vase, *etc*.

References

[1] Sid Yingze Bao, Manmohan Chandraker, Yuanqing Lin, and Silvio Savarese. Dense object reconstruction with semantic priors. In *CVPR*, 2013. 2

[2] Volker Blanz and Thomas Vetter. A morphable model for the synthesis of 3d faces. In *SIGGRAPH*, 1999. 2

[3] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *CVPR*, 2017. 1

[4] Angel X. Chang, Thomas A. Funkhouser, Leonidas J. Guibas, Pat Hanrahan, Qi-Xing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. Shapenet: An information-rich 3d model repository. In *CoRR 1512.03012*, 2015. 1, 2, 5

[5] Christopher B. Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3D-R2N2: A unified approach for single and multi-view 3d object reconstruction. In *ECCV*, 2016. 1, 2, 5

[6] Amaury Dame, Victor A. Prisacariu, Carl Y. Ren, and Ian Reid. Dense reconstruction using 3d object shape priors. In *CVPR*, 2013. 2

[7] Mathieu Desbrun, Mark Meyer, Peter Schröder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH*, 1999. 5

[8] Haoqiang Fan, Hao Su, and Leonidas J. Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, 2017. 1, 2, 5, 6

[9] Sanja Fidler, Sven Dickinson, and Raquel Urtasun. 3d object detection and viewpoint estimation with a deformable 3d cuboid model. In *NeurIPS*, 2012. 2

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

[11] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. In *IJRR*, 2013. 2

[12] Ross Girshick. Fast R-CNN. In *ICCV*, 2015. 2

[13] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 1, 2

[14] Benjamin Graham, Martin Engelcke, and Laurens van der Maaten. 3d semantic segmentation with submanifold sparse convolutional networks. In *CVPR*, 2018. 2

[15] Thibault Groueix, Matthew Fisher, Vladimir G Kim, Bryan C Russell, and Mathieu Aubry. A papier-mâché approach to learning 3d surface generation. In *CVPR*, 2018. 2

[16] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from rgb-d images for object detection and segmentatio. In *ECCV*, 2014. 2

[17] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 2

[18] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2, 3

[19] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[20] Derek Hoiem, Alexei A. Efros, and Martial Hebert. Geometric context from a single image. In *ICCV*, 2005. 2

[21] Christian Hne, Nikolay Savinov, and Marc Pollefeys. Class specific 3d object shape priors using surface normals. In *CVPR*, 2014. 2

[22] Max Jaderberg, Karen Simonyan, and Andrew Zisserman. Spatial transformer networks. In *NeurIPS*, 2015. 4

[23] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018. 1, 2

[24] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. In *NeurIPS*, 2017. 2

[25] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *CVPR*, 2018. 2, 5

[26] Alex Kendall, Hayk Martirosyan, Saumitro Dasgupta, Peter Henry, Ryan Kennedy, Abraham Bachrach, and Adam Bry. End-to-end learning of geometry and context for deep stereo regression. In *ICCV*, 2017. 2

[27] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[28] Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017. 4

[29] Alex Krizhevsky, Ilya Sutskever, and Geoff Hinton. ImageNet classification with deep convolutional neural networks. In *NeurIPS*, 2012. 1

[30] Abhijit Kundu, Yin Li, and James M. Rehg. 3d-rcnn: Instance-level 3d object reconstruction via render-and-compare. In *CVPR*, 2018. 2

[31] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *NeurIPS*, 2018. 2

[32] Joseph J. Lim, Hamed Pirsiavash, and Antonio Torralba. Parsing IKEA Objects: Fine Pose Estimation. In *ICCV*, 2013. 2

[33] Chen-Hsuan Lin, Chen Kong, and Simon Lucey. Learning efficient point cloud generation for dense 3d object reconstruction. In *AAAI*, 2018. 2

[34] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017. 3, 8

[35] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017. 2

[36] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollr, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. *ECCV*, 2014. 1, 2

[37] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, and Scott Reed. SSD: Single shot multibox detector. In *ECCV*, 2016. 2

[38] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. 3

[39] William E. Lorensen and Harvey E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. In *SIGGRAPH*. ACM, 1987. 4

[40] Priyanka Mandikal, Navaneet Murthy, Mayank Agarwal, and R. Venkatesh Babu. 3d-lmnet: Latent embedding matching for accurate and diverse 3d point cloud reconstruction from a single image. In *BMVC*, 2018. 2

[41] Lars Mescheder, Michael Oechsle, Michael Niemeyer, Sebastian Nowozin, and Andreas Geiger. Occupancy networks: Learning 3d reconstruction in function space. In *CVPR*, 2019. 2, 5

[42] George A. Miller. Wordnet: A lexical database for english. In *Commun. ACM*, 1995. 5

[43] Georgios Pavlakos, Xiaowei Zhou, Aaron Chan, Konstantinos G. Derpanis, and Kostas Daniilidis. 6-dof object pose from semantic keypoints. In *ICRA*, 2017. 2

[44] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *NeurIPS*, 2017. 2

[45] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *CVPR*, 2016. 2

[46] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015. 1, 2, 3

[47] Danilo Jimenez Rezende, S.M. Ali Eslami, Shakir Mohamed, Peter Battaglia, Max Jaderberg, and Nicolas Heess. Unsupervised learning of 3d structure from images. In *NeurIPS*, 2016. 2

[48] Stephan R. Richter and Stefan Roth. Matryoshka networks: Predicting 3d geometry via nested shape layers. In *CVPR*, 2018. 2

[49] Gernot Riegler, Ali Osman Ulusoy, and Andreas Geiger. Octnet: Learning deep 3d representations at high resolutions. In *CVPR*, 2017. 2

[50] Jason Rock, Tanmay Gupta, Justin Thorsen, JunYoung Gwak, Daeyun Shin, and Derek Hoiem. Completing 3d object shape from one depth image. In *CVPR*, 2015. 2

[51] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, and Li Fei-Fei. ImageNet Large Scale Visual Recognition Challenge. *IJCV*, 2015. 1, 2

[52] Daniel Scharstein and Richard Szeliski. A taxonomy and evaluation of dense two-frame stereo correspondence algorithms. *IJCV*, 2002. 2

[53] Tanner Schmidt, Richard Newcombe, and Dieter Fox. Self-supervised visual descriptor learning for dense correspondence. In *IEEE Robotics and Automation Letters*, 2017. 2

[54] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. In *ICLR*, 2015. 1

[55] Edward Smith, Scott Fujimoto, and David Meger. Multi-view silhouette and depth decomposition for high resolution 3d object representation. In *NeurIPS*, 2018. 2, 5

[56] Edward J. Smith, Scott Fujimoto, Adriana Romero, and David Meger. GEOMetrics: Exploiting geometric structure for graph-encoded objects. In *ICML*, 2019. 1, 2, 4, 5

[57] Shuran Song and Jianxiong Xiao. Deep Sliding Shapes for amodal 3D object detection in RGB-D images. In *CVPR*, 2016. 2

[58] Hang Su, Varun Jampani, Deqing Sun, Subhransu Maji, Evangelos Kalogerakis, Ming-Hsuan Yang, and Jan Kautz. SPLATNet: Sparse lattice networks for point cloud processing. In *CVPR*, 2018. 2

[59] Xingyuan Sun, Jiajun Wu, Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Tianfan Xue, Joshua B. Tenenbaum, and William T. Freeman. Pix3d: Dataset and methods for single-image 3d shape modeling. In *CVPR*, 2018. 2, 5, 7

[60] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *CVPR*, 2015. 1

[61] Maxim Tatarchenko, Alexey Dosovitskiy, and Thomas Brox. Octree generating networks: Efficient convolutional architectures for high-resolution 3d outputs. In *ICCV*, 2017. 2

[62] Maxim Tatarchenko, Jaesik Park, Vladlen Koltun, and Qian-Yi Zhou. Tangent convolutions for dense prediction in 3d. In *CVPR*, 2018. 2

[63] Yonglong Tian, Andrew Luo, Xingyuan Sun, Kevin Ellis, William T. Freeman, Joshua B. Tenenbaum, and Jiajun Wu. Learning to infer and execute 3d shape programs. In *ICLR*, 2019. 2

[64] Alexander Toshev and Christian Szegedy. Deeppose: Human pose estimation via deep neural networks. In *CVPR*, 2014. 1

[65] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *CVPR*, 2015. 2

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

[67] Shubham Tulsiani, Tinghui Zhou, Alexei A. Efros, and Jitendra Malik. Multi-view supervision for single-view reconstruction via differentiable ray consistency. In *CVPR*, 2017. 2

[68] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2Mesh: Generating 3D mesh models from single RGB images. In *ECCV*, 2018. 1, 2, 4, 5, 6, 7

[69] Peng-Shuai Wang, Chun-Yu Sun, Yang Liu, and Xin Tong. Adaptive O-CNN: a patch-based deep representation of 3d shapes. In *SIGGRAPH Asia*, 2018. 2

[70] Jiajun Wu, Yifan Wang, Tianfan Xue, Xingyuan Sun, Bill Freeman, and Josh Tenenbaum. Marrnet: 3d shape reconstruction via 2.5 d sketches. In *NeurIPS*, 2017. 2

[71] Jiajun Wu, Chengkai Zhang, Tianfan Xue, Bill Freeman, and Josh Tenenbaum. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling. In *NeurIPS*, 2016. 2

[72] Jiajun Wu, Chengkai Zhang, Xiuming Zhang, Zhoutong Zhang, William T. Freeman, and Joshua B. Tenenbaum. Learning 3D Shape Priors for Shape Completion and Reconstruction. In *ECCV*, 2018. 2

[73] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *WACV*, 2014. 2

[74] Xinchen Yan, Jimei Yang, Ersin Yumer, Yijie Guo, and Honglak Lee. Perspective transformer nets: Learning single-view 3d object reconstruction without 3d supervision. In *NeurIPS*, 2016. 2

[75] Xiuming Zhang, Zhoutong Zhang, Chengkai Zhang, Joshua B. Tenenbaum, William T. Freeman, and Jiajun Wu. Learning to Reconstruct Shapes from Unseen Classes. In *NeurIPS*, 2018. 2