Learning 3D Object Shape and Layout without 3D Supervision

Georgia Gkioxari[1] Nikhila Ravi[1] Justin Johnson[1,2]

[1]Meta AI [2]University of Michigan

Abstract

A 3D scene consists of a set of objects, each with a shape and a layout giving their position in space. Understanding 3D scenes from 2D images is an important goal, with applications in robotics and graphics. While there have been recent advances in predicting 3D shape and layout from a single image, most approaches rely on 3D ground truth for training which is expensive to collect at scale. We overcome these limitations and propose a method that learns to predict 3D shape and layout for objects without any ground truth shape or layout information: instead we rely on multi-view images with 2D supervision which can more easily be collected at scale. Through extensive experiments on ShapeNet, Hypersim, and ScanNet we demonstrate that our approach scales to large datasets of realistic images, and compares favorably to methods relying on 3D ground truth. On Hypersim and ScanNet where reliable 3D ground truth is not available, our approach outperforms supervised approaches trained on smaller and less diverse datasets. [1]

1. Introduction

A 3D scene consists of a set of objects, specified by a 3D *shape* for each object and the 3D *layout* of objects in space. Understanding this 3D scene structure is critical for navigating or interacting with the world. Unfortunately, directly measuring or perceiving 3D structure is often impractical. For this reason, inferring the shape and layout of 3D scenes from 2D images has long been a fundamental problem in computer vision, with wide applications in robotics, autonomous vehicles, graphics, AR/VR, and beyond.

The rise of deep learning has dramatically improved 3D understanding from a single image. Methods have advanced from estimating 3D shapes of isolated objects [6, 11, 51] to predicting multiple shapes in complex scenes [14] and even jointly predicting shape and layout [38, 50]. While impressive, these methods share a flaw: they use ground truth 3D shape and layout for training. Creating large, varied training sets with this data is impractical, limiting the scalability and utility of methods relying on strong 3D supervision.

[1]Project page https://gkioxari.github.io/usl/

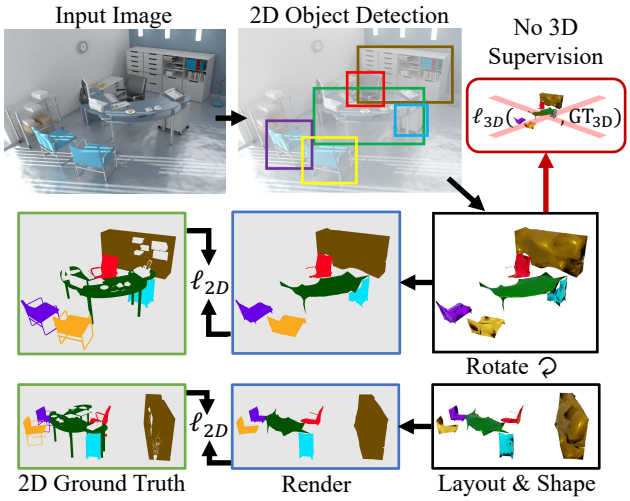

Figure 1. We propose an end-to-end model which takes an input image, detects all objects in 2D and predicts their 3D shapes and layouts. We learn from multiple 2D scene views, *e.g.* frames of videos, and without any 3D supervision.

Some recent approaches take an extreme position, and train on collections of images without any 3D supervision whatsoever [15, 22, 27, 28, 53]. While admirable, overcoming the fundamental ambiguities of 3D from a single image requires strong category-specific shape priors, making it difficult to scale to the complexities of the real world.

Another natural way to predict 3D structure is to use multiple views. Multi-view images give weak 3D supervision and can be captured at scale using videos or multi-camera rigs. Classical techniques such as Structure from Motion and Multi-View Stereo [18] reconstruct full 3D scenes without 3D supervision, but require many views, do not predict semantics, and are not typically learned from data.

More recently, *differentiable rendering* has enabled a new wave of methods that predict 3D shapes without strong 3D supervision [5, 24, 34, 41]. During training a model inputs a single image and outputs a 3D shape, which is rendered from one or more auxiliary views; comparing the rendered prediction with 2D silhouettes in auxiliary views provides a training signal. This pipeline is promising since it requires no ground truth 3D shapes, instead learning solely from multi-view images and 2D image supervision which can

both be collected at scale. However, to date this technique has only been applied to simple images with a single object.

In this paper, we aim to predict 3D object shapes and layout in complex scenes from a single image, as in Fig. 1. Crucially we do not use ground truth shapes or layouts during training; instead we learn from object silhouettes in multi-view images. We build on Mesh R-CNN [14], which predicts 3D shapes, but not layouts, for objects in complex scenes and relies on 3D shape supervision during training. We augment Mesh R-CNN with a *layout network* that estimates each object's 3D location, and replace expensive mesh supervision with scalable multi-view supervision. Like prior work [5, 24, 34, 41] we learn via differentiable rendering and 2D losses; however these methods only predict 3D shape – to also predict layout we use a *distance transform loss*. We call our **U**nsupervised approach for **S**hape and **L**ayout estimation USL. At test time, USL inputs a single RGB image and jointly detects objects and predicts their 3D shape and layout.

We show results on three datasets to demonstrate the utility of our scalable multi-view supervision. First, we show results on Scene-ShapeNet, a synthetic dataset with scenes composed of multiple ShapeNet [3] objects where our method shows strong performance compared with Mesh R-CNN trained using strong 3D shape supervision. We then experiment on Hypersim [44], demonstrating that our approach scales to complex, realistic scenes with many objects. Finally, we show results on ScanNet [7] where camera poses are estimated from BundleFusion [8] and 2D silhouettes are estimated using PointRend [26], showing that we can learn from noisy real-world video without expensive ground truth.

2. Related Work

3D scene and object reconstruction from multiple posed views has been studied extensively, from traditional Structure from Motion (SfM) and Multi-View Stereo (MVS) [18, 45], aided by shape priors [1, 2, 9, 21] to learning-based techniques [23, 25, 46]. These methods require multiple views at test time. In this work, we focus on single image inference.

Data driven methods predict object shape and layout from a single image of a novel scene. [16] predicts 3D object boxes from RGB-D inputs. [50] combines oriented 3D object boxes with canonical voxel shapes from RGB inputs; Total3D [38] replaces voxels with meshes. Mesh R-CNN [14] predicts 3D object meshes via intermediate voxel predictions but does not tackle layout. All these methods are supervised with 3D annotations, via 3D bounding boxes [37, 48] or 3D object shapes, *e.g.* CAD models [3, 49]. However, 3D annotations are costly and involve complicated annotation pipelines, limiting their availability to few object and scene types. Our work shares the same goal with the above methods; we predict 3D object shapes and layouts in view coordinates from a single image, but we do so *without 3D supervision*.

For the task of shape prediction, weakly supervised methods eliminate the need for 3D annotations by using category specific object priors [15, 22, 27, 28, 30, 55], or 2D keypoints [22, 40]. While these methods show promising results for a select few object types, their ability to scale to more classes is questionable. In this work, we do not use object priors which allows us to scale to many more object categories. Complementing shape, we predict object layouts, namely object positions in 3D, in an end-to-end manner.

A natural way to eliminate the need for 3D supervision and object priors is to learn from multiple views. Differentiable rendering [5, 24, 29, 34, 35, 39, 41] allows information to flow to 3D from 2D re-projections. [5, 17, 24, 34, 41] achieve object reconstruction from a single view via re-projection from 2 or more views during training. [53] adversarially compare silhouette re-projections with objects from an image collection. While ground breaking, these methods focus on images of single objects in simple settings, *e.g.* on a white background. In this work, we use differentiable rendering to learn from multiple views. However, we focus on realistic multi-object scenes, which pose significant challenges stemming from occlusion and ambiguity from multiple instances.

Recent methods train neural networks to predict pixel-wise depth from a single image, using video frames [36, 56] or 3D supervision [4, 10, 31, 54]. While related to layout, pixel-wise depth only captures the visible parts of the objects and commonly normalized depth is predicted. In this work, our goal is to reconstruct complete object shapes and predict their 3D location in metric space, *e.g.* in meters.

3. Method

Our model inputs a single RGB image, detects objects, and for each detected object outputs a 3D *shape* (triangle mesh) and *layout* (position in 3D space). Taken together, these outputs make up a full 3D scene as shown in Figure 1.

During training, our model is not supervised by any ground truth 3D shape or layout information. These annotations are expensive, so relying upon them severely limits the scale of available training data. Instead, we use multiple RGB views of each scene together with 2D ground truth: 3D shapes predicted from one view are differentiably rendered from other views, where their 2D silhouettes are compared with 2D ground truth silhouettes.

We build on Mesh R-CNN [14], which extends Mask R-CNN [19] to jointly detect objects and predict 3D shapes. We make three major modifications to Mesh R-CNN. First, we use a new mechanism for computing vertex-aligned features called *RoIMap* which better preserves aspect ratio information and improves 3D shape prediction. Second, we introduce an additional *layout head* for predicting the 3D position of each object. Third (and most importantly), we eliminate the need for 3D shape supervision, instead learning with 2D supervision from multiple views.

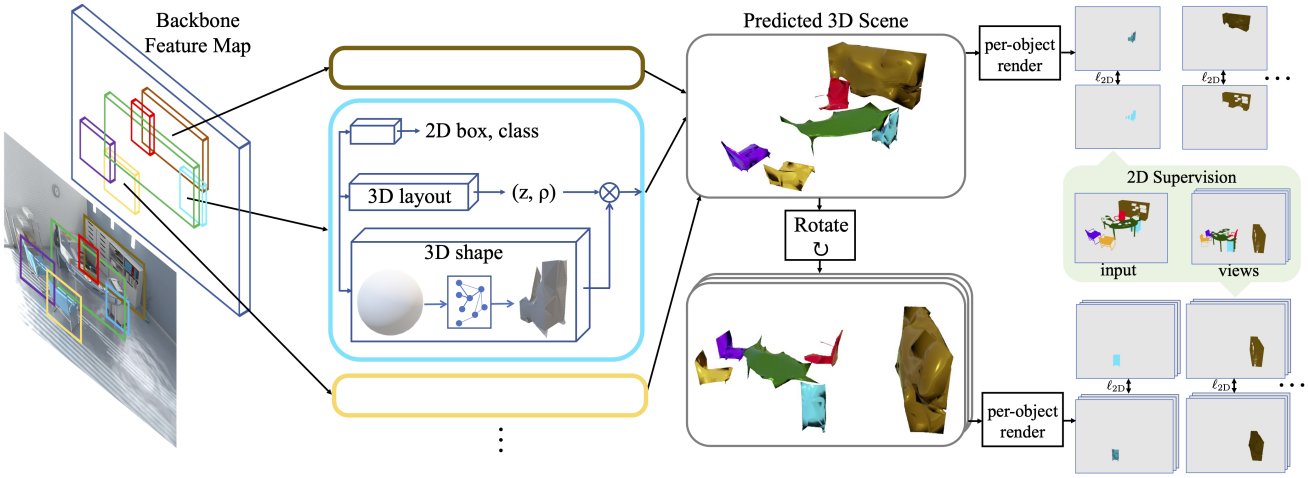

Figure 2. Our model takes as input an RGB image, detects all objects in 2D and predicts their 3D location and shape via *layout* and *shape* heads, respectively. The output is a scene composed of all detected 3D objects. During training, the scene is differentiably rendered from other views and compared with the 2D ground truth. We use no 3D shape or layout supervision.

3.1. Method Overview

The architecture of our system largely follows Mesh R-CNN [14], modified to allow training without ground truth shape or layout. An overview can be seen in Figure 2.

The input image is first processed by a *backbone network* (ResNet50-FPN [20, 32] in our experiments) which extracts a *backbone feature map*. A *region proposal network* (RPN) [42] then gives category-agnostic *regions of interest* (RoIs) which are processed by task-specific *heads*. The *box head* performs 2D recognition; following [19] it predicts a 2D bounding box and semantic category per RoI. The *layout head* performs 3D localization: for each RoI it predicts depth extent and 3D position of the object's center. The *shape head* predicts a 3D triangle mesh for each RoI; following [14, 51] it deforms an initial sphere mesh via graph convolutions.

The box and layout heads receive input from the backbone network via *RoIAlign* [19] which crops and resizes regions from the backbone feature map. The shape head receives per-vertex features from the backbone network via *RoIMap*.

During training, we assume access to M views of the scene with camera poses and instance segmentations. The model takes as input the first view and predicts the 3D scene, which is differentiably rendered from all M views and compared with the 2D ground truth.

3.2. Layout Prediction

Our model predicts a 3D position for each object, parameterized as an axis-aligned box with a 3D center and length along each coordinate axis. The box head localizes objects *in the image plane*; this relies on direct image evidence, since marking the pixels belonging to each object suffices for 2D localization. In contrast, scale/depth ambiguity makes localization *vertical to the image plane* hard from image evidence alone, and must rely on prior knowledge about the world.

We thus use a separate *layout head* to localize objects in depth. It predicts each object's length ρ along the depth axis and the depth z of its center. RoI features from RoIAlign [19] are average-pooled and passed to an MLP which predicts scalars $\tilde{\rho}, \tilde{z} \in (0, 1)$ via a sigmoid. Then

$$\rho = \rho_0 + \tilde{\rho}(\rho_1 - \rho_z) \qquad z = z_0 + \tilde{z}(z_1 - z_0) \qquad (1)$$

where $\{\rho_0, \rho_1, z_0, z_1\}$ are dataset-specific hyperparamters setting the minimum and maximum object depth and extent. Like Mask R-CNN's box head, predictions for $\tilde{\rho}$ and $\tilde{z}$ are category-specific so the model can learn per-category priors.

3.3. Shape Prediction

For each detected object, the *shape head* outputs a 3D triangle mesh $\mathcal{T} = (V, F)$ with vertices V and faces F. Predictions compose the 3D scene and are not 3D supervised.

We follow Mesh R-CNN's [14] mesh refinement branch, which deforms an initial mesh $\mathcal{T}_0 = (V_0, F)$ via a sequence of S *mesh refinement stages*, each comprising three operations: *feature sampling* gives an image-aligned feature for each vertex; *graph convolution* propagates information along mesh edges; and *vertex refinement* predicts offsets dV_i for each vertex and updates vertex positions $V_i = V_{i-1} + dV_i$. The final stage's output gives the predicted shape: $V = V_S$.

Mesh R-CNN predicts a voxelized shape for each object, giving rise to instance-specific initial meshes. This requires 3D voxel supervision and cannot be used in our setting; we thus use an identical sphere for each object's initial mesh.

To enable equivariance to 3D translation, we predict shapes for each object in a normalized space with V_0 and each dV_i in the range $[-1, +1]$. Predicted shapes are cast into the 3D scene using a pinhole camera model: the $[-1, +1]^3$ cube in normalized space is mapped via a homography to the object frustum defined by the camera intrinsics and the outputs of the box and layout heads (see Figure 3).

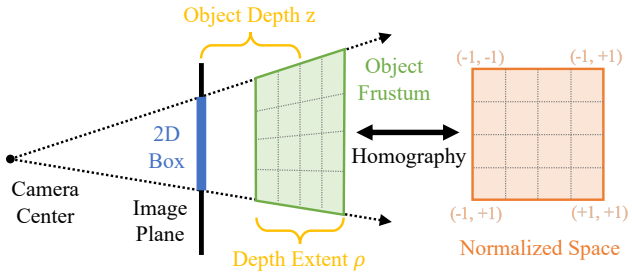

Figure 3. We cast shape predictions into the 3D scene by combining outputs from all heads. The box head predicts a 2D box and the layout head predicts object depth z and depth extent ρ; together with camera intrinsics these define an object frustum in 3D space. The shape head predicts a mesh in normalized space which is mapped to the object frustum via a homography.

RoIMap. The shape head must precisely localize each vertex in 3D. To this end, each mesh refinement stage receives features from the backbone by projecting the current mesh onto the image plane and bilinearly interpolating to sample a feature aligned to each vertex. Though conceptually simple, the exact mechanism for sampling affects performance.

Mesh R-CNN [14] uses RoIAlign [19] to compute a fixed-sized feature map per RoI, then uses VertAlign to sample vertex features from the RoI features (see Fig. 4). This causes several issues. First, RoI features are a fixed square size, so vertex features do not respect the aspect ratio in the input image. Second, repeated bilinear interpolation (first by RoIAlign then VertAlign) may cause artifacts. Third, features cannot be computed for verticies outside the RoI.

As shown in Fig. 4, we overcome these issues by sampling vertex features directly from the backbone feature map rather than from RoI features. We call this approach *RoIMap*. Our experiments in Sec. 4 show that this seemingly small change significantly improves overall performance. A similar approach was used in [26] for instance segmentation.

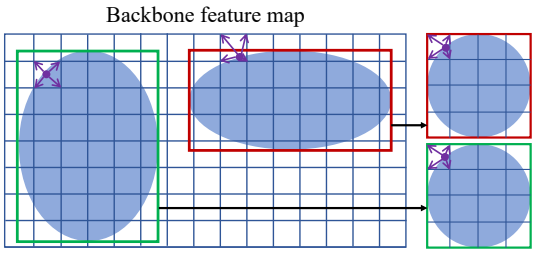

Figure 4. Mesh R-CNN [14] samples vertex features from per-RoI features computed via RoIAlign. We instead use *RoIMap* which samples vertex features directly from the backbone feature map.

3.4. Learning without 3D Supervision

We assume that ground truth for 3D object shapes and layouts is expensive and thus cannot be used to directly supervise the shape and layout heads. Instead, we supervise our model using only 2D ground truth from multiple views.

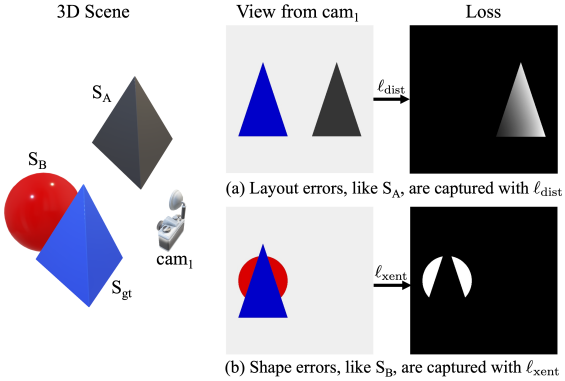

Figure 5. Erros in (a) layout and (b) shape dictate the appropriate loss functions for comparing image silhouettes.

During training we sample M RGB views $\{I_1, \ldots, I_M\}$ of a scene with known poses, so $R_{i\to j}$ transforms 3D points in the camera view of I_i to that of I_j. Let $\mathcal{O}$ be the set of objects visible in I_1, and S_j^o be the ground-truth silhouette of $o \in \mathcal{O}$ in I_j. Our model inputs I_1 and predicts 3D shapes $\{\mathcal{T}_1^o\}_{o\in\mathcal{O}}$ in the camera view of I_1. We compute predicted 2D silhouettes from *all views*, $\hat{S}_{1\to j}^o = \text{render}(R_{1\to j} \cdot \mathcal{T}_1^o)$, using a differentiable silhouette renderer [34, 41]. Our training loss for learning 3D shape and layout is then

$$\mathcal{L}_{\text{3D}} = \frac{1}{|\mathcal{O}|}\frac{1}{M}\sum_{o\in\mathcal{O}}\sum_{j=1}^{M}\ell_{\text{2D}}(\hat{S}_{1\to j}^o, S_j^o) \quad (2)$$

where ℓ_{2D} compares a pair of 2D masks using separate terms to correct errors in *shape* and *layout*.

As shown in Fig. 5, a pixel-wise cross-entropy loss ℓ_{xent} gives a useful learning signal when two masks overlap but differ in *shape*. However when there is no overlap ℓ_{xent} equally penalizes all predictions and thus does not tell the model how to correct errors in *layout*.

We thus introduce a *distance transform loss* ℓ_{dist} which penalizes 2D distances between masks:

$$\ell_{\text{dist}}(\hat{S}, S) = \int_{\hat{S}} \inf_{s\in S} \|\hat{s} - s\|_2^2 \, d\hat{s} + \int_{S} \inf_{\hat{s}\in\hat{S}} \|\hat{s} - s\|_2^2 \, ds. \quad (3)$$

This loss penalizes non-overlapping silhouettes depending on their screen-space distance, aiding layout prediction. We approximate ℓ_{dist} as a bidirectional Chamfer distance between points sampled from $\hat{S}$ and S. To sample from $\hat{S}$, we sample from the surface of the mesh (like [47]) then project onto the image plane, so ℓ_{dist} does not require computing $\hat{S}$.

Distance transforms have a long history in object detection [12, 13]; a similar loss was used by [22] to learn texture.

Our overall loss on pairs of 2D silhouettes is thus

$$\ell_{\text{2D}}(\hat{S}, S) = \ell_{\text{dist}}(\hat{S}, S) + \mathbf{1}[IoU(\hat{S}, S)>0.5] \cdot \ell_{\text{xent}}(\hat{S}, S) \quad (4)$$

only applying ℓ_{xent} for overlapping silhouettes (IoU>0.5).

Our full training loss is a weighted combination of Mask R-CNN's 2D losses, our 3D loss $\mathcal{L}_{\text{3D}}$, and 3D shape regularizers encouraging smooth mesh predictions.

Model	3D Metrics Ch.(↓)	3D Metrics F_1	$Mask_{2D}$ IoU Input	$Mask_{2D}$ IoU Views
Fixed depth	0.275	20.1	18.4	14.1
Random depth	0.202	23.3	21.5	17.6
USL$^{(2)}$	0.050	62.9	**53.6**	43.4
USL$^{(5)}$	**0.034**	**70.9**	52.3	**46.7**
USL$^{(5)}$ w/o RoIMap	0.059	55.1	51.1	40.9
USL$^{(5)}$ w/o ℓ_{dist}	0.039	68.9	42.9	37.1
Mesh R-CNN [14]	**0.015**	**87.9**	**61.5**	**57.7**

Table 1. Performance on Scene-ShapeNet val. We report a *random* and a *fixed* depth baseline which place a sphere for each object at random and a fixed depth, respectively, We report our model, USL, trained with 2 & 5 views and ablate RoIMap and ℓ_{dist}. We compare to Mesh R-CNN [14] which is the supervised state-of-the-art.

Dynamic rendering. Objects in real-world scenes tend to occupy few image pixels, so naively computing $\hat{S}^o_{1\to j}$ spends significant resources rasterizing pixels not occupied by objects, limiting rendering resolution. We thus use a *dynamic rendering* scheme: when computing $\hat{S}^o_{1\to j}$, we only render a region which is the union of the ground truth silhouette and the projection of the predicted mesh $\mathcal{T}^o_1$ onto view j. This allows rendering at 4× resolution vs naive rendering, which captures finer object details and improves results.

4. Experiments

We experiment on three datasets: Scene-ShapeNet, Hypersim [44] and ScanNet [7]. Scene-ShapeNet forms simple scenes from ShapeNet [3] objects, while Hypersim and ScanNet contain video sequences of complex scenes with multiple objects under varying appearance, occlusion and lighting conditions; a stark difference to single object benchmarks.

Scene-ShapeNet provides ground truth 3D shape and layout, enabling comparison with supervised methods and the use of 3D evaluation metrics. 3D ground truth is not available on Hypersim and ScanNet, so we resort to proxy metrics by comparing rendered predictions to 2D ground truth from multiple views. We perform extensive quantitative analysis and show predictions on challenging images of novel scenes. Compared to state-of-the-art supervised methods trained on smaller and less diverse 3D annotated datasets, we show that our method can better generalize to novel scenes.

Metrics. In the absence of 3D ground truth, we use an evaluation scheme which relies on multi-view 2D comparisons.

Specifically, we project each object's predicted 3D shape to all available views of the scene. In each view we compute the intersection-over-union (IoU) between the rendered prediction and ground truth object mask in that view. We report two metrics: *$Mask_{2D}$ IoU input* is the IoU in the view the model receives as input, and *$Mask_{2D}$ IoU views* is the mean across all other views; both are averaged over all scenes.

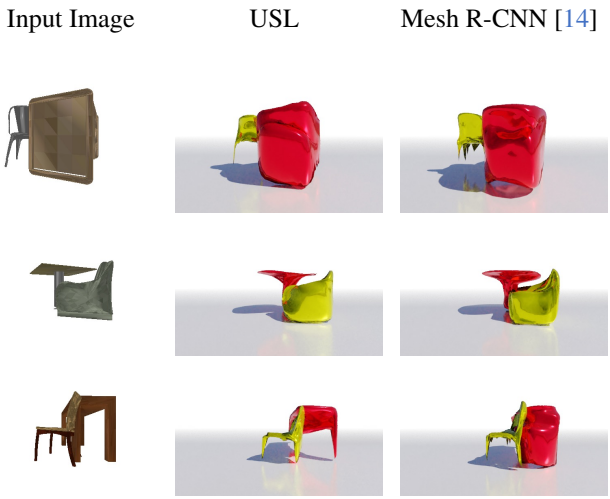

Figure 6. Predictions on Scene-ShapeNet val. We show the input image (left) and the predicted 3D objects and layout of our USL (middle) and 3D supervised Mesh R-CNN [14] (right).

4.1. Results on Scene-ShapeNet

We introduce Scene-ShapeNet, a dataset of scenes formed of ShapeNet [3] objects. Its scenes contain object pairs from three object types, namely *chair, sofa* and *table*. Objects are placed at random 3D locations and poses and scenes are rendered from multiple viewpoints. The dataset consists of 86.4k images and 4k unique object shapes, split into 80%/10%/10% for train/test/val. *Each split contains unique object models and scenes*. See Appendix for more details. This dataset provides 3D ground truth, so we can evaluate predicted shapes and layouts in 3D.

Training details. We follow Mesh R-CNN [14] and train using Adam for 25 epochs with batches of 64 images on 8 V100 GPUs. For each example in the batch, we randomly sample M views from the corresponding scene. Input images are 512×512; we render with PyTorch3D [41] at a resolution of 128×128 with 10 faces per pixel; blur radius and blend sigma are 10^{-3}. The backbone ResNet50 is pretrained on ImageNet; other parameters are learned from scratch.

Evaluation. We evaluate on val, which contains objects and scenes disjoint from train. Since 3D ground truth is available, in addition to $Mask_{2D}$ IoU we also report standard 3D metrics: 3D chamfer distance and F_1@0.1m, following [14].

Results are shown in Table 1. We compare to variants of USL with $M \in \{2, 5\}$ views and ablate RoIMap and the distance transform loss, ℓ_{dist}. We report a *random* baseline which predicts each object as a sphere with random depth $z \in [1.4, 2.0]$ and depth extent $\rho \in [0.1, 1.0]$, and a *fixed-depth* baseline which predicts each object as sphere with $z = 1.7, \rho = 1.0$. We also compare to Mesh R-CNN [14] trained on Scene-ShapeNet with full 3D supervision.

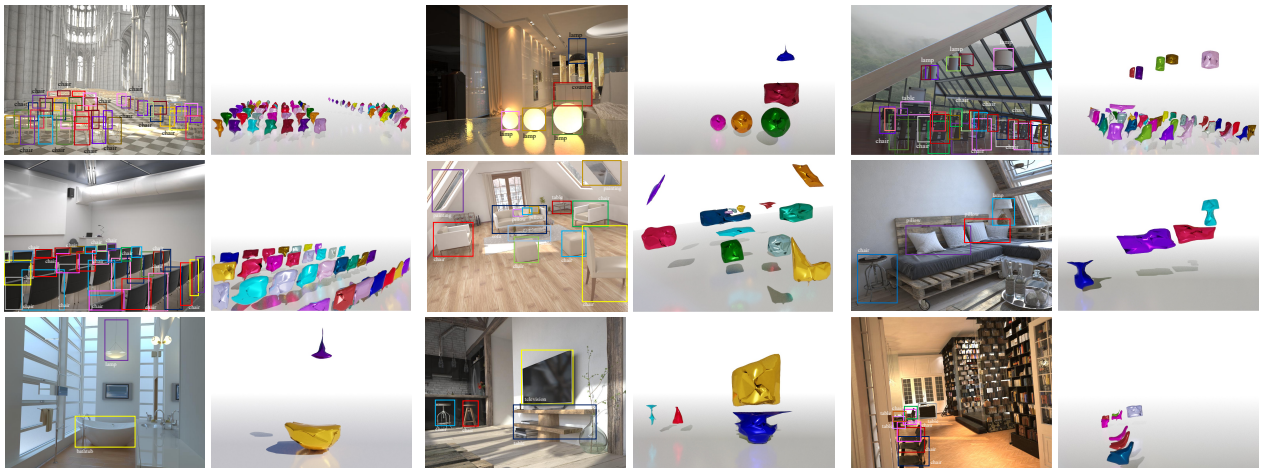

Figure 7. Predictions on Hypersim. For each example, we show the input image and the detected 2D objects (left) and the predicted 3D objects and layout (right). Examples are of complex scenes with many objects of diverse appearances and types. See our video animations.

From Table 1, Mesh R-CNN performs the best as is expected (7th row). Our USL(5) performs best among all unsupervised baselines (4th row). We observe that performance drops when we replace RoIMap with RoIAlign (5th row) and when we omit ℓ_{dist} (6th row). A model trained with 2 views (3rd row) performs worse than the 5-view model. Finally, we note that our Mask_{2D} IoU on views correlates with 3D metrics, validating its choice as a proxy for 3D performance.

Figure 6 compares predictions from USL(5) and Mesh R-CNN on Scene-ShapeNet. Though USL receives no 3D supervision, it predicts accurate 3D layouts and shapes even when objects are occluded. Mesh R-CNN makes more accurate predictions (see Table 1) but requires 3D supervision, which is expensive to obtain at scale. See Appendix for more qualitative examples.

4.2. Results on Hypersim

We experiment on Hypersim [44], a dataset of 461 complex scenes, each rendered along camera trajectories giving 77,400 images with ground truth pose, semantic and instance masks for 40 object categories, and instance IDs linking objects across views. Hypersim has an average of 50 instances and 10 object types per image; in contrast, COCO [33] images have just 7 instances on average.

Training details. We train on Hypersim's train split of 365 scenes. We follow the Mask R-CNN [19] recipe [52], training with batches of 16 images on 8 V100 GPUs for 80k iterations. We use SGD with initial learning rate 10^{-2}, decaying by 0.1 after 66k and 74k iterations. For each example in the batch, we randomly sample M views from the corresponding video. We use PyTorch3D [41] and render dynamically at 72×72, 10 faces per pixel, blur radius and blend sigma of 10^{-3}. The backbone ResNet50-FPN is pretrained on COCO; all other parameters are learned from scratch. More details can be found in the Appendix.

Evaluation. We evaluate on Hypersim val, which consists of 46 scenes disjoint from train. Hypersim does not provide ground truth 3D object shape and layout information so we cannot report 3D metrics. However, it provides ground truth pixel-wise metric depth (in meters) which when combined with instance masks gives ground truth depth on the visible parts of each object. This allows us to additionally report a *Depth* L_1 metric (for both the input and other views) between the true and predicted nearest depth of each object. Finally, we also report *Box$_{2D}$ gIoU* on input and views, by computing the 2D image-aligned boxes bounding the predicted and true object silhouettes, used in *Mask$_{2D}$ IoU*, and measure the gIoU [43], a generalization of traditional IoU which measures the proximity between two boxes.

Results. We compare to a *random* baseline which predicts each object as a sphere at random depth $z\in[1.0, 10.0]$ and depth extent $\rho\in[0.1, 1.0]$, and a *fixed-depth* baseline, which predicts objects as spheres fixed at $z=5$, $\rho=0.5$. We train *layout-only* variants with $M\in\{2, 5\}$ views, which learn *layout* but predict *shape* as fixed spheres. We train USL with $M\in\{2, 5\}$ views for both 3D layout and shape.

Finally, we compare to Mesh R-CNN [14]. Hypersim does not release public 3D shape annotations needed to train Mesh R-CNN, so we instead train Mesh R-CNN on Pix3D as in [14]. Pix3D only provides ground truth *shape* but not *layout*, so Mesh R-CNN trained on Pix3D can only predict shape. We thus combine *shapes* predicted by Mesh R-CNN with *layouts* predicted by USL for this baseline.

Table 2 shows the performance on val. To ensure fair comparisons we decouple 3D understanding from 2D detection by using ground truth 2D boxes on the input image during evaluation for all baselines (Box_{2D} gIoU = 1.0 for input). Notably, our model's 2D object detector, trained jointly with the shape and layout networks, achieves an AP of 64% and AP^{50} of 73%. We report our model's performance when using its own object detections in the last row of Table 2.

Model	Predicts		Box_{2D} gIoU		Mask_{2D} IoU		Depth L_1 (↓)	
	Layout	Shape	Input	Views	Input	Views	Input	Views
Random depth	✗	✗	1.00	0.13	0.58	0.20	3.55	3.28
Fixed depth	✗	✗	1.00	0.22	0.58	0.24	2.73	2.59
Layout-Only$^{(2)}$	✓	✗	1.00	0.25	0.58	0.25	2.51	2.35
Layout-Only$^{(5)}$	✓	✗	1.00	**0.37**	0.58	0.30	1.81	1.74
USL$^{(2)}$	✓	✓	1.00	0.30	0.73	0.33	1.90	1.83
USL$^{(5)}$	✓	✓	1.00	0.33	**0.74**	**0.34**	**1.78**	**1.72**
Mesh R-CNN [14] + USL$^{(5)}$ layout	✓	✓	1.00	0.21	0.36	0.22	**1.78**	**1.72**
USL$^{(5)}$ w/ detections	✓	✓	0.92	0.31	0.72	0.33	1.80	1.74

Table 2. Results on Hypersim val. We report a *random* and a *fixed-depth* baseline which place a sphere at random and a fixed depth, respectively, for each object. We train *Layout-Only* variants of our approach with 2 and 5 views which represent objects as spheres at the predicted 3D locations. Finally, we train our USL with 2 and 5 views which learn both shape and layout. We report the performance of Mesh R-CNN [14] pretrained on Pix3D [49] for 3D shape prediction and combined with USL$^{(5)}$ layout predictions. The final row shows the performance of USL$^{(5)}$ when using the model's object predictions instead of ground truth detections.

	Mask_{2D} IoU		Depth L_1	
	Input	Views	Input	Views
RoIAlign	0.67	0.20	4.38	3.86
RoIMap	**0.74**	**0.34**	**1.78**	**1.72**

(a)

ℓ_{dist}	Mask_{2D} IoU		Depth L_1	
	Input	Views	Input	Views
✗	0.74	0.28	2.61	2.44
✓	0.74	**0.34**	**1.78**	**1.72**

(b)

Model	Mask_{2D} IoU	
	Input	Views
Sphere-Only	0.58	0.45
USL$^{(5)}$	**0.74**	**0.53**

(c)

Table 3. Ablations on Hypersim for (a) *RoIAlign* vs. *RoIMap*, (b) distance transform loss ℓ_{dist}, and with (c) oracle depth.

From Table 2, our USL$^{(5)}$ model (6$^{\text{th}}$ row) outperforms the *layout-only*, *random* and *fixed-depth* baselines for Mask_{2D} IoU and Depth L_1. The *layout-only*$^{(5)}$ baseline has a higher Box_{2D} gIoU but a lower Mask_{2D} IoU on views than USL$^{(5)}$ (4$^{\text{th}}$ vs. 6$^{\text{th}}$ row), indicating that it works well for layout but not for shape. Training with 5 views is better than 2 views (5$^{\text{th}}$ vs. 6$^{\text{th}}$ and 3$^{\text{rd}}$ vs. 4$^{\text{th}}$ row), which is expected as 5 views during training provide more information. Using more than 5 views does not improve performance further, likely because walk-through videos cap the number of frames with new information about each part of the scene. The Mesh R-CNN baseline achieves low performance despite being supervised, proving that existing 3D annotated datasets are insufficient and don't generalize well to more complex scenes. Finally, we observe that Mask_{2D} IoU correlates with Depth L_1 on views, as models with higher IoU have lower depth error. In the absence of any 3D ground truth, Mask_{2D} IoU is likely to serve as a good proxy for 3D metrics.

Table 3a compares RoIMap to RoIAlign and shows the impact of RoIMap on the performance. Table 3b ablates the distance transform ℓ_{dist} (Equation 3) which proves crucial to our model's performance. Finally, note that Mask_{2D} IoU on views captures both shape and layout errors; wrong layout predictions even for accurate shapes can result in low IoU. To decouple performance for layout and shape, we compare our USL$^{(5)}$ to a *sphere-only* baseline, which represents each object as a sphere, and provide true object depth for both models at test time in Table 3c. From Table 3c we see that we outperform *sphere-only* for shape.

Figure 7 shows predictions on Hypersim for diverse novel scenes with many object instances and types, including *lamp*, *painting*, *sofa*, *chair*, *table*, *tv*, *bathtub* and *counter*. We observe that our model captures layout well, while object shapes are roughly correct but certainly less refined. Predicting detailed 3D shapes without 3D supervision is hard. In addition to the lack of 3D supervision, we learn from views extracted from walk-through videos which capture scenes from a constrained, far from 360^o, set of views (*e.g.* backs of couches are never seen, etc.). This is in contrast to SceneShapeNet, where 360^o scene views are available, and thus our model is able to capture shape more accurately.

Comparison to Total3D. We compare to Total3D [38], a state-of-the-art fully supervised method for predicting shape and layout from a single image. Total3D learns a *layout model* on SUN RGB-D [48] which provides oriented 3D object bounding boxes, and learns a *shape model* on Pix3D [49] which provides image aligned CAD models for 9 object classes. At test time, predictions from the shape model are positioned according to predictions from the layout model; this gives final predictions in *view coordinates*.

Figure 8 qualitatively compares to Total3D on randomly selected input images; see more in the Appendix. Despite being supervised, we observe that Total3D tends to fetch the nearest shape for the object class, which does not match the object's appearance in the input. For example, in the 1$^{\text{st}}$ example it predicts a rectangular table instead of a round one shown in the image. Regarding layout, Total3D struggles to place objects in the correct relative locations, resulting in large shape intersections and erroneous layouts. We also note that the 3D objects do not align with the 2D objects (2$^{\text{nd}}$ vs 4$^{\text{rd}}$ col); Total3D does not enforce alignment with 2D

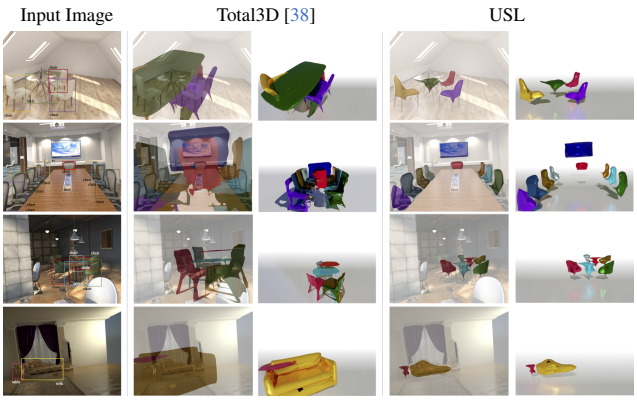

Figure 8. Comparison of Total3D [38] and our approach. The input image is shown in the 1st col. We show the predicted 3D shapes and layout perspectively projected into the image plane along with a 3D visualization of the predicted layout, for Total3D (2nd & 3rd col) and our approach (4th & 5th col). More in the Appendix.

contrary to our approach which does so by design (Figure 3). Figure 8 proves that Total3D has an extremely hard time generalizing to complex scenes, despite being supervised with 3D ground truth for shape and layout. We draw the same conclusion when comparing to Mesh R-CNN in Table 2. This is further proof that existing 3D annotated datasets, like Pix3D and SUN RGB-D, are not adequate. Training on larger more diverse datasets could improve Total3D's performance, but 3D annotations are expensive to collect at large scales. Our approach is a first attempt to tackle 3D object layout on complex scenes bypassing the need for 3D supervision.

4.3. Results on ScanNet

We experiment on ScanNet [7], a dataset of videos of indoor scenes with reconstructed cameras. We detect object instances and object tracks by applying a PointRend [26] model pre-trained on COCO [33]. COCO consists of 80 object categories out of which more than 30 are detected on ScanNet, including *bottle*, *keyboard*, *books*, *potted plant*, *tv* and more. We track objects with a simple mask IoU heuristic and keep tracks of at least 50 frames to ensure diversity of object views for training and evaluation. This results in 210k instances across 160k frames and more than 2700 tracks.

Training details. We train on the ScanNet training set for 200k iterations, with a learning rate of 0.01 which drops by 0.1 at 150k & 170k iterations. During training, we sample M views from the object tracks with a probability proportional to the time distance from the input to encourage diverse views. We initialize the backbone with COCO-pretrained weights, and learn the remaining parameters from scratch.

Results. Table 4 shows results on ScanNet val, which consists of scenes distinct from train. We evaluate a *random* and *fixed-depth* baseline and our USL trained with 5 views.

Figure 9. Predictions on ScanNet val. We show the input image along with detected 2D objects (top) and the predicted 3D objects and layout from our approach (bottom).

	Predicts		Box_{2D} gIoU		Mask_{2D} IoU	
Model	L	S	Input	Views	Input	Views
Random depth	✗	✗	1.00	0.36	0.73	0.39
Fixed depth	✗	✗	1.00	0.34	0.73	0.38
$\text{USL}^{(5)}$	✓	✓	1.00	**0.61**	**0.84**	**0.61**

Table 4. Results on ScanNet val. We compare our model trained with 5 views, $\text{USL}^{(5)}$, to a *random* and *fixed-depth* baseline.

We report Box_{2D} gIoU and Mask_{2D} IoU for the input and views. For fair comparisons, we feed the same input boxes detected from PointRend to all models. For the view metrics, we evaluate on the furthest 20% frames in the track from each input view to put emphasis on diverse object views. From Table 4, our model outperforms all baselines.

Figure 9 shows predictions on ScanNet. We show the input image with 2D object detections superimposed (top) and the predicted 3D object layouts (bottom) for a variety of scenes. ScanNet is challenging as frames have motion blur and are narrow views of scenes with heavy object truncations by the image border. Yet, our model is able to reason about 3D objects and their location in the scene.

5. Discussion

This paper presents a method for predicting 3D object shape and layout from a single image. We learn without ground truth shapes and layouts, and instead rely on multiple views with 2D annotations. Our experiments on three datasets show compelling results on images of novel scenes with many objects. We compare to fully supervised methods and prove our model's superiority to generalize to complex scenes. On complex scenes at test time, we notice that our model captures layout well but shapes are of lower quality. This is not a surprise as supervision comes from 2D silhouettes and thus our 3D shapes cannot easily capture fine details. Texture information could improve shapes, which we leave for future work. While our work is a first attempt at learning scene layouts and 3D object shapes from videos end-to-end and without 3D supervision, there is still a lot to be done for models to work in the wild for thousands of object classes and all types of real world scenes.

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] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, et al. Shapenet: An information-rich 3d model repository. *arXiv preprint arXiv:1512.03012*, 2015. 2, 5

[4] Weifeng Chen, Zhao Fu, Dawei Yang, and Jia Deng. Single-image depth perception in the wild. In *NeurIPS*, 2016. 2

[5] Wenzheng Chen, Huan Ling, Jun Gao, Edward Smith, Jaakko Lehtinen, Alec Jacobson, and Sanja Fidler. Learning to predict 3d objects with an interpolation-based differentiable renderer. In *NeurIPS*, 2019. 1, 2

[6] 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

[7] Angela Dai, Angel X. Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 2, 5, 8

[8] Angela Dai, Matthias Nießner, Michael Zollhöfer, Shahram Izadi, and Christian Theobalt. Bundlefusion: Real-time globally consistent 3d reconstruction using on-the-fly surface reintegration. *ToG*, 2017. 2

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

[10] David Eigen, Christian Puhrsch, and Rob Fergus. Depth map prediction from a single image using a multi-scale deep network. In *NeurIPS*, 2014. 2

[11] 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

[12] Pedro F Felzenszwalb, Ross B Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *PAMI*, 2009. 4

[13] Pedro F Felzenszwalb and Daniel P Huttenlocher. Distance transforms of sampled functions. *Theory of computing*, 2012. 4

[14] Georgia Gkioxari, Jitendra Malik, and Justin Johnson. Mesh R-CNN. In *ICCV*, 2019. 1, 2, 3, 4, 5, 6, 7

[15] Shubham Goel, Angjoo Kanazawa, , and Jitendra Malik. Shape and viewpoints without keypoints. In *ECCV*, 2020. 1, 2

[16] Saurabh Gupta, Pablo Arbelaez, and Jitendra Malik. Perceptual organization and recognition of indoor scenes from rgb-d images. In *CVPR*, 2013. 2

[17] Zhizhong Han, Chao Chen, Yu-Shen Liu, and Matthias Zwicker. Drwr: A differentiable renderer without rendering for unsupervised 3d structure learning from silhouette images. *arXiv preprint arXiv:2007.06127*, 2020. 2

[18] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003. 1, 2

[19] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 2, 3, 4, 6

[20] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 3

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

[22] Angjoo Kanazawa, Shubham Tulsiani, Alexei A Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018. 1, 2, 4

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

[24] Hiroharu Kato, Yoshitaka Ushiku, and Tatsuya Harada. Neural 3D mesh renderer. In *CVPR*, 2018. 1, 2

[25] 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

[26] Alexander Kirillov, Yuxin Wu, Kaiming He, and Ross Girshick. PointRend: Image segmentation as rendering. In *CVPR*, 2020. 2, 4, 8

[27] Nilesh Kulkarni, Abhinav Gupta, David Fouhey, and Shubham Tulsiani. Articulation-aware canonical surface mapping. In *CVPR*, 2020. 1, 2

[28] Nilesh Kulkarni, Abhinav Gupta, and Shubham Tulsiani. Canonical surface mapping via geometric cycle consistency. In *ICCV*, 2019. 1, 2

[29] Tzu-Mao Li, Miika Aittala, Frédo Durand, and Jaakko Lehtinen. Differentiable monte carlo ray tracing through edge sampling. *TOG*, 2018. 2

[30] Xueting Li, Sifei Liu, Kihwan Kim, Shalini De Mello, Varun Jampani, Ming-Hsuan Yang, and Jan Kautz. Self-supervised single-view 3d reconstruction via semantic consistency. In *ECCV*, 2020. 2

[31] Zhengqi Li and Noah Snavely. Megadepth: Learning single-view depth prediction from internet photos. In *CVPR*, 2018. 2

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

[33] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft COCO: Common objects in context. In *ECCV*, 2014. 6, 8

[34] Shichen Liu, Weikai Chen, Tianye Li, and Hao Li. Soft rasterizer: Differentiable rendering for unsupervised single-view mesh reconstruction. In *ICCV*, 2019. 1, 2, 4

[35] Matthew M Loper and Michael J Black. OpenDR: An approximate differentiable renderer. In *ECCV*, 2014. 2

[36] Xuan Luo, Jia-Bin Huang, Richard Szeliski, Kevin Matzen, and Johannes Kopf. Consistent video depth estimation. *TOG*, 2020. 2

[37] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2

[38] Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Jun Zhang. Total3dunderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In *CVPR*, 2020. 1, 2, 7, 8

[39] Merlin Nimier-David, Delio Vicini, Tizian Zeltner, and Wenzel Jakob. Mitsuba 2: a retargetable forward and inverse renderer. *TOG*, 2019. 2

[40] David Novotny, Nikhila Ravi, Benjamin Graham, Natalia Neverova, and Andrea Vedaldi. C3dpo: Canonical 3d pose networks for non-rigid structure from motion. In *ICCV*, 2019. 2

[41] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 1, 2, 4, 5, 6

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

[43] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union. In *CVPR*, 2019. 6

[44] 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. In *ICCV*, 2021. 2, 5, 6

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

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

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

[48] Shuran Song, Samuel P Lichtenberg, and Jianxiong Xiao. Sun rgb-d: A rgb-d scene understanding benchmark suite. In *CVPR*, 2015. 2, 7

[49] 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, 7

[50] Shubham Tulsiani, Saurabh Gupta, David Fouhey, and Alexei A. Efrosand Jitendra Malik. Factoring shape, pose, and layout from the 2d image of a 3d scene. In *CVPR*, 2018. 1, 2

[51] 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, 3

[52] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 6

[53] Yufei Ye, Shubham Tulsiani, and Abhinav Gupta. Shelf-supervised mesh prediction in the wild. In *CVPR*, 2021. 1, 2

[54] Wei Yin, Jianming Zhang, Oliver Wang, Simon Niklaus, Long Mai, Simon Chen, and Chunhua Shen. Learning to recover 3d scene shape from a single image. In *CVPR*, 2021. 2

[55] Jason Y Zhang, Sam Pepose, Hanbyul Joo, Deva Ramanan, Jitendra Malik, and Angjoo Kanazawa. Perceiving 3d human-object spatial arrangements from a single image in the wild. In *ECCV*, 2020. 2

[56] Tinghui Zhou, Matthew Brown, Noah Snavely, and David G Lowe. Unsupervised learning of depth and ego-motion from video. In *CVPR*, 2017. 2