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

Appendix

In this appendix we provide additional information about our approach. We provide details about the model's architecture and the training recipe and add more qualitative results. Video animations which complement our visualizations in the paper can be found in our project's page https://gkioxari.github.io/usl/.

A. Comparison to Total3D

Figure 1 shows more examples comparing Total3D [5] and our approach. In both the Supplementary and in the main paper, we compare on examples which cover a variety of object and scene types which allows us to better assess generalization of both methods to diverse scenes. From Figure 1 we notice that when projecting Total3D predictions on the image plane the renderings are not aligned to the input image (2nd col.). This is a consequence of Total3D's model design which regresses to 3D object boxes and shapes without guaranteed alignment with the image. In contrast, our predictions are by design aligned to the input image (4th col.). In addition and as noted in the main paper, Total3D has a difficult time placing objects in correct configurations leading to wrong layout and often to large shape intersections (3rd col.). In contrast, our approach, even though not trained with 3D supervision, produces more accurate layout predictions (5th col.). Finally, we notice that Total3D tends to fetch the nearest shape in semantic space for each object which is commonly not an accurate representation for the particular instance of that object, *e.g.* the ottoman chairs in the 3rd example which have a cuboid shape are represented as chairs with four legs and a back (placed under the coffee table). Our approach more accurately represents object shapes, even if not fine in detail.

B. Experiments on Scene-ShapeNet

We create Scene-ShapeNet, a dataset of scenes composed of synthetic 3D objects. We provide details and sample images of the Scene-ShapeNet dataset. We show more results comparing to Mesh R-CNN [3], trained with 3D supervision on the Scene-ShapeNet training set.

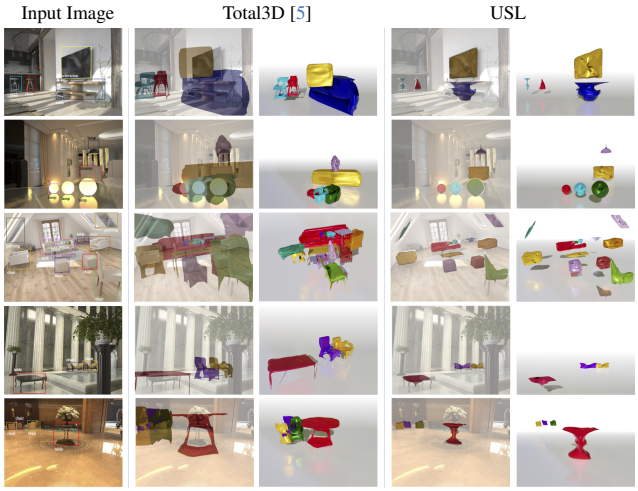

Figure 1. Additional comparisons of Total3D [5] and our approach. The input image is shown in the 1st col. We show the predicted 3D shapes and layout perspectively projected into the image plane along with a 3D visualization of the predicted layout, for Total3D (2nd & 3rd col) and our approach (4th &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

B.1. The dataset

As mentioned in the paper, we create the Scene-ShapeNet dataset from synthetic ShapeNet [1] objects by pairing object instances from the {*chair, table, sofa*} categories to create scenes. Specifically, for each scene, we randomly choose a model for an object type. Each model is randomly rotated around the Y axis ("up"), $\theta_Y \in [0^o, 360^o]$, and their center is placed at a random (X, Z) location on the $Y = 0$ plane such that $Z \in [1.5, 1.9]$ and $X \in [-0.4, 0.4]$. We render the scene by rotating a camera around the objects at multiple azimuth angles and heights. We don't exclude scenes with intersecting objects to make the task more challenging as it leads to bigger occlusions, a characteristic of the real world which makes recognition difficult. Figure 3 shows image examples from the dataset. Each row shows two views of the same scene. The spatial configuration of the pair of objects and their 3D shapes vary across the dataset.

Figure 2. 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 [3] (right).

B.2. More results

Figure 2 shows more comparisons of our USL and Mesh R-CNN [3] on images from Scene-ShapeNet val. Mesh R-CNN is trained with 3D supervision on the Scene-ShapeNet training set, while USL does not use 3D supervision. Mesh R-CNN predicts more accurate shapes, as expected. Even though our USL was not trained with 3D supervision, its predictions are quite good both for estimating the layout and the 3D object shapes. This is also validated by our quantitative analysis in Table 1 in the main paper, where our USL performs competitively. Animations of our predictions are shown in https://gkioxari.github.io/usl/.

B.3. Network architecture and stats

For the Scene-ShapeNet experiments we follow exactly the same architecture as Mesh R-CNN [3] for ShapeNet, without the voxel head as we don't have 3D supervision, and the same training recipe. Refer to [3] for more details.

Loss. We define the training objective as $\mathcal{L} = \mathcal{L}_{\text{3D}} + \mu_{\text{reg}} \cdot \mathcal{L}_{\text{reg}}$. We set $\mu_{\text{reg}} = 0.1$ while $\mathcal{L}_{\text{reg}} = \mathcal{L}_{\text{edge}}$ is an edge length regularizer, as in [3].

Training stats. We distribute training across 8 V100s with a total of 64 images per batch (8 images per batch per GPU). For our 5-view model, each iteration takes 3.4sec and consumes 7.3GB of memory per GPU. Inference runs at 8fps.

C. Experiments on Hypersim

C.1. Video Animations

Animations of our 3D shape and layout predictions on Hypersim val and test can be found in https://gkioxari.github.io/usl/.

C.2. Network architecture and stats

Table 3 shows the network architecture used for our model on Hypersim [6]. We skip the architecture of the RPN and box head, as they are identical to Mask R-CNN [4].

Loss. We define the training objective as $\mathcal{L} = \mathcal{L}_{\text{2D}} + \mu_{\text{3D}} \cdot \mathcal{L}_{\text{3D}} + \mu_{\text{reg}} \cdot \mathcal{L}_{\text{reg}}$. We set $\mu_{\text{3D}} = 1.0$, $\mu_{\text{reg}} = 0.05$ while $\mathcal{L}_{\text{reg}} = \frac{1}{2}||dV||_2^2$ is a simple L2 regularizer on the predicted deformations dV (before the `tanh` layer). We found that a L2 regularizer performs similarly to $\mathcal{L}_{\text{edge}}$ but the L2 is more efficient allowing us to scale to many object detections, as is the case in Hypersim images.

Training stats. We distribute training across 8 V100s with a total of 16 images per batch (2 images per batch per GPU). For our 5-view model, each iteration takes about 1.1 sec and consumes 6.1GB of memory per GPU. We train for 80k iterations. During inference, our model runs at 3fps.

C.3. Performance on the test set.

For all of our ablations and experiments in the main paper we train on the Hypersim training set and evaluate on the Hypersim validation set. Table 2 shows results of our USL$^{(5)}$ on the Hypersim test set with a model trained with 5 views on the Hypersim trainval set.

D. Experiments on ScanNet

D.1. Network architecture and stats

Table 4 shows the network architecture used for our model on ScanNet [2]. We again skip the RPN and box head, as they are identical to Mask R-CNN [4].

Loss. We define the training objective as $\mathcal{L} = \mathcal{L}_{\text{2D}} + \mu_{\text{3D}} \cdot \mathcal{L}_{\text{3D}} + \mu_{\text{reg}} \cdot \mathcal{L}_{\text{reg}}$. We set $\mu_{\text{3D}} = 1.0$ and $\mu_{\text{reg}} \cdot \mathcal{L}_{\text{reg}} = 0.05 \cdot \frac{1}{2}||dV||_2^2 + 0.1 \cdot \mathcal{L}_{\text{laplac}}$, where $\mathcal{L}_{\text{laplac}}$ is a Laplacian regularizer. The laplacian regularizer is added for the ScanNet dataset because the object views are not as diverse compared to Hypersim. The laplacian regularizer encourages shapes to be smooth and discourages them from degenerating; though is not critical and a similar result could have been achieved perhaps if we increased the weight of the L2 regularizer.

Training stats. We distribute training across 8 V100s with a total of 16 images per batch (2 images per batch per GPU). For our 5-view model, each iteration takes about 0.7 sec and 3.2GB of memory per GPU. We train for 200k iterations.

Table 1 summarizes the differences in the training recipes across the three datasets. Note that these differences are

motivated by (a) fair comparison with prior work and (b) memory/runtime. On Scene-ShapeNet our optimizer, backbone, initialization, and regularization are chosen to match the ShapeNet experiments from Mesh R-CNN [3]. Changing these to match Hypersim / ScanNet would not significantly affect results, but hinders fair comparison with [3]. The choice of regularizer is not critical; all perform similarly. We use $\mathcal{L}_{edge}$ on ShapeNet to match [3], but found that L2 gives similar results with reduced runtime; adding Laplacian on ScanNet gives slightly smoother meshes but is not critical. We vary training iterations and LR decay schedule with dataset size. 3 vs 1 refinement stage marginally improves results, but is slower (1.1s vs 0.7s per iteration); due to ScanNet's large size we use 1 stage for faster experiments.

	Scene-ShapeNet	Hypersim	ScanNet
Backbone	R50	R50-FPN	R50-FPN
Initialization	ImageNet	COCO	COCO
Optimizer / LR	Adam / 10^{-3}	SGD / 10^{-2}	SGD / 10^{-2}
Regularization	$\mathcal{L}_{\text{edge}}$	$\mathcal{L}_2$	$\mathcal{L}_2$ + $\mathcal{L}_{\text{laplac}}$
Render Resolution	128	72	72
Refinement Stages	3	3	1
Training Iterations	30K	80K	200K

Table 1. Recipe differences between datasets.

Figure 3. Sample images from the Scene-ShapeNet dataset. Each row contains an example from two different views. Images are of scenes of object pairs under random spatial configurations.

	Predicts		Box_{2D} gIoU		Mask_{2D} IoU		Depth L_1 (↓)	
Model	Layout	Shape	Input	Views	Input	Views	Input	Views
USL[(5)]	✓	✓	1.00	0.35	0.69	0.36	2.07	1.98
USL[(5)] w/ detections	✓	✓	0.93	0.34	0.68	0.35	2.08	2.00

Table 2. Performance of our USL[(5)] on the Hypersim test set trained with 5 views on Hypersim trainval, with ground truth input boxes (1st row) and the model's object detections (2nd row).

Index	**Inputs**	**Operation**	**Output shape**
(1)	Input	Input Image	$H \times W \times 3$
(2)	(1)	Backbone: ResNet-50-FPN p2 level	$\frac{H}{4} \times \frac{W}{4} \times 256$
(3)	(2)	RoIAlign: Pools and Averages RoI features	1×256
(4)	(3)	$4\times$ Linear(256, 256)	1×256
(5)	(4)	Layout z: Linear(256, $\lvert\mathcal{C}\rvert$)	$1 \times \lvert\mathcal{C}\rvert$
(6)	(4)	Layout ρ: Linear(256, $\lvert\mathcal{C}\rvert$)	$1 \times \lvert\mathcal{C}\rvert$
(7)	(2)	Stage 1: RoIMap: Pools features from ico3 sphere	$\lvert V\rvert \times 256$, $\lvert F\rvert \times 3$
(8)	(7)	Stage 1: $3\times$ GraphConv(256 + 3, 256)	$\lvert V\rvert \times 256$, $\lvert F\rvert \times 3$
(9)	(8)	Stage 1: Linear(256 + 3, 3)	$\lvert V\rvert \times 3$, $\lvert F\rvert \times 3$
(10)	(2), (9)	Stage 2: RoIMap: Pools features from (9)	$\lvert V\rvert \times 256$, $\lvert F\rvert \times 3$
(11)	(10)	Stage 2: $3\times$ GraphConv(256 + 3, 256)	$\lvert V\rvert \times 256$, $\lvert F\rvert \times 3$
(12)	(11)	Stage 2: Linear(256 + 3, 3)	$\lvert V\rvert \times 3$, $\lvert F\rvert \times 3$
(13)	(2), (12)	Stage 3: RoIMap: Pools features from (12)	$\lvert V\rvert \times 256$, $\lvert F\rvert \times 3$
(14)	(13)	Stage 3: $3\times$ GraphConv(256 + 3, 256)	$\lvert V\rvert \times 256$, $\lvert F\rvert \times 3$
(15)	(14)	Stage 3: Linear(256 + 3, 3)	$\lvert V\rvert \times 3$, $\lvert F\rvert \times 3$

Table 3. Overall architecture of our model on Hypersim. The backbone, RPN and box branches are identical to Mask R-CNN [4]. The RPN produces a bounding box prediction for anchors at each spatial location in the input feature map; a subset of these candidate boxes are processed by the other branches, but here we show only the shapes resulting from processing a single box for the subsequent task-specific heads. Here, $|\mathcal{C}|$ is the number of categories.

Index	**Inputs**	**Operation**	**Output shape**
(1)	Input	Input Image	$H \times W \times 3$
(2)	(1)	Backbone: ResNet-50-FPN p2 level	$\frac{H}{4} \times \frac{W}{4} \times 256$
(3)	(2)	RoIAlign: Pools and averages RoI features	1×256
(4)	(3)	$4\times$ Linear(256, 256)	1×256
(5)	(4)	Layout C_Z: Linear(256, $\lvert\mathcal{C}\rvert$)	$1 \times \lvert\mathcal{C}\rvert$
(6)	(4)	Layout ρ_Z: Linear(256, $\lvert\mathcal{C}\rvert$)	$1 \times \lvert\mathcal{C}\rvert$
(7)	(2)	Stage 1: RoIMap: Pools features from ico3 sphere	$\lvert V\rvert \times 256$, $\lvert F\rvert \times 3$
(8)	(7)	Stage 1: $3\times$ GraphConv(256 + 3, 256)	$\lvert V\rvert \times 256$, $\lvert F\rvert \times 3$
(9)	(8)	Stage 1: Linear(256 + 3, 3)	$\lvert V\rvert \times 3$, $\lvert F\rvert \times 3$

Table 4. Overall architecture of our model on ScanNet. The backbone, RPN and box branches are identical to Mask R-CNN [4]. The RPN produces a bounding box prediction for anchors at each spatial location in the input feature map; a subset of these candidate boxes are processed by the other branches, but here we show only the shapes resulting from processing a single box for the subsequent task-specific heads. Here, $|\mathcal{C}|$ is the number of categories.

References

[1] 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. 1

[2] 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

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

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

[5] 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

[6] 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