OMNI3D: A Large Benchmark and Model for 3D Object Detection in the Wild Appendix

Garrick Brazil[1] Abhinav Kumar[2] Julian Straub[1] Nikhila Ravi[1]
Justin Johnson[1] Georgia Gkioxari[3]
[1]Meta AI [2]Michigan State University [3]Caltech

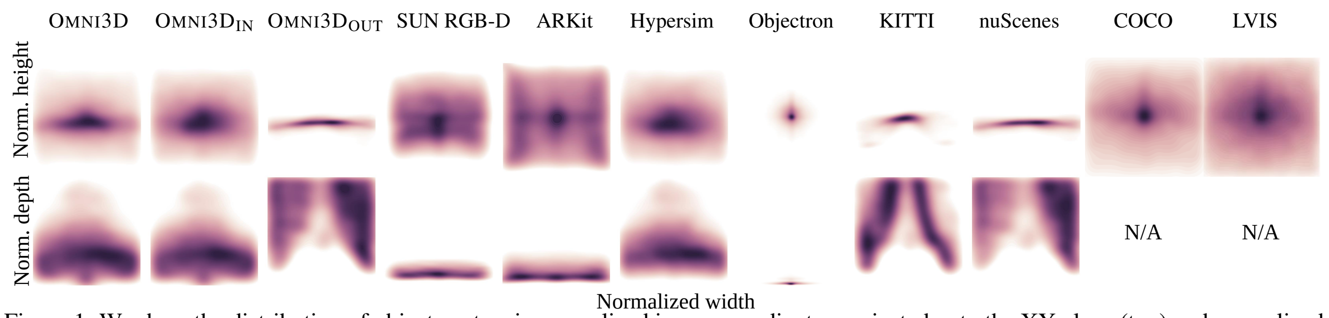


Figure 1. We show the distribution of object centers in normalized image coordinates projected onto the XY-plane (top) and normalized depth projected onto the topview XZ-plane (bottom) across each subset of OMNI3D, individual 3D datasets SUN RGB-D [17], ARKit [2], Hypersim [15], Objectron [1], KITTI [6], nuScenes [4], and large-scale 2D datasets COCO [11] and LVIS [7].

1. Dataset Details

In this section, we provide details related to the creation of OMNI3D, its sources, coordinate systems, and statistics.

Sources. For each individual dataset used in OMNI3D, we use their official train, val, and test set for any datasets which have all annotations released. If no public test set is available then we use their validation set *as* our test set. Whenever necessary we further split the remaining training set in 10:1 ratio by sequences in order form train and val sets. The resultant image make up of OMNI3D is detailed in Table 1. To make the benchmark coherent we merged semantic categories across the sources, *e.g.*, there are 26 variants of chair including 'chair', 'chairs', 'recliner', 'rocking chair', *etc.* We show the category instance counts in Figure 2.

Coordinate system. We define our unified 3D coordinate system for all labels with the camera center being the origin and +x facing right, +y facing down, +z inward [6]. Object pose is relative to an initial object with its bottom-face normal aligned to +y and its front-face aligned to +x (*e.g.* upright and facing to the right). All images have known camera calibration matrices with input resolutions varying from 370 to 1920 and diverse focal lengths from 518 to 1708 in pixels. Each object label contains a category label, a 2D bounding box, a 3D centroid in camera space meters, a 3×3 matrix defining the object to camera rotation, and the physical dimensions (width, height, length) in meters.

Method	$\lvert C\rvert$	$\lvert C^*\rvert$	Total	Train	Val	Test
KITTI [6]	8	5	7,481	3,321	391	3,769
SUN RGB-D [17]	83	38	10,335	4,929	356	5,050
nuScenes [4]	9	9	34,149	26,215	1,915	6,019
Objectron [1]	9	9	46,644	33,519	3,811	9,314
ARKitScenes [2]	15	14	60,924	48,046	5,268	7,610
Hypersim [15]	32	29	74,619	59,543	7,386	7,690
OMNI3D$_{\text{OUT}}$	14	11	41,630	29,536	2,306	9,788
OMNI3D$_{\text{IN}}$	84	38	145,878	112,518	13,010	20,350
OMNI3D	98	50	234,152	175,573	19,127	39,452

Table 1. We detail the statistics for each dataset split. We report the total number of categories $\lvert C\rvert$, and the number of categories $\lvert C^*\rvert$ used in our paper and our AP$_{\text{3D}}$ metrics. C^* contains all categories from C with at least 1000 positive instances. Finally, we report the total number of images split into train/val/test.

Spatial Statistics. Following Section 3 of the main paper, we provide the spatial statistics for the individual data sources of SUN RGB-D [17], ARKit [2], Hypersim [15], Objectron [1], KITTI [6], nuScenes [4], as well as OMNI3D$_{\text{IN}}$ and OMNI3D$_{\text{OUT}}$ in Figure 1. As we mention in the main paper, we observe that the indoor domain data, with the exception of Hypersim, have bias for close objects. Moreover, outdoor data tend to be spatially biased with projected centroids along diagonal ground planes while indoor is more central. We provide a more detailed view of density-normalized depth distributions for each dataset in Figure 3.

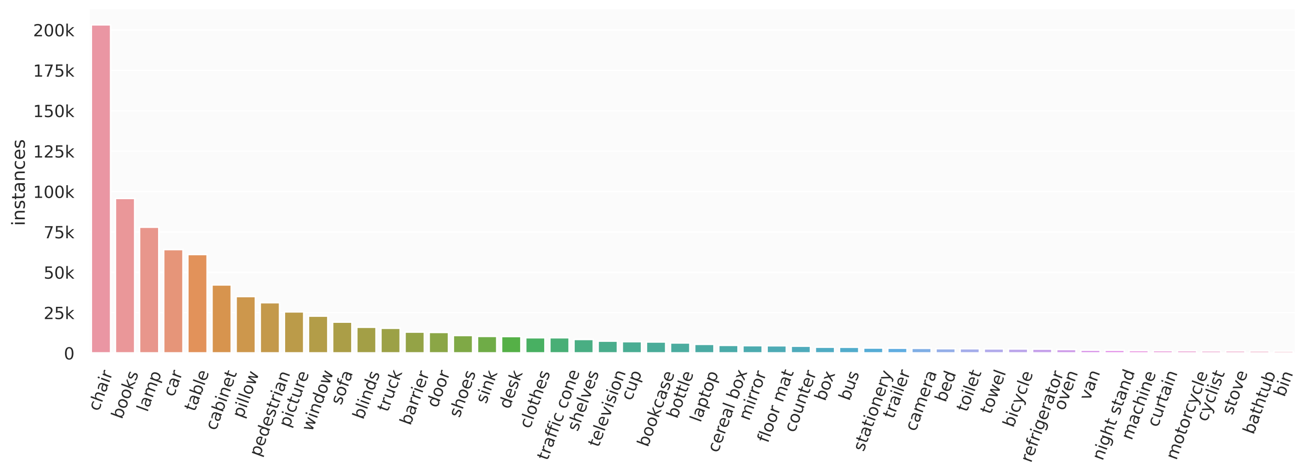


Figure 2. Number of instances for the 50 categories in OMNI3D.

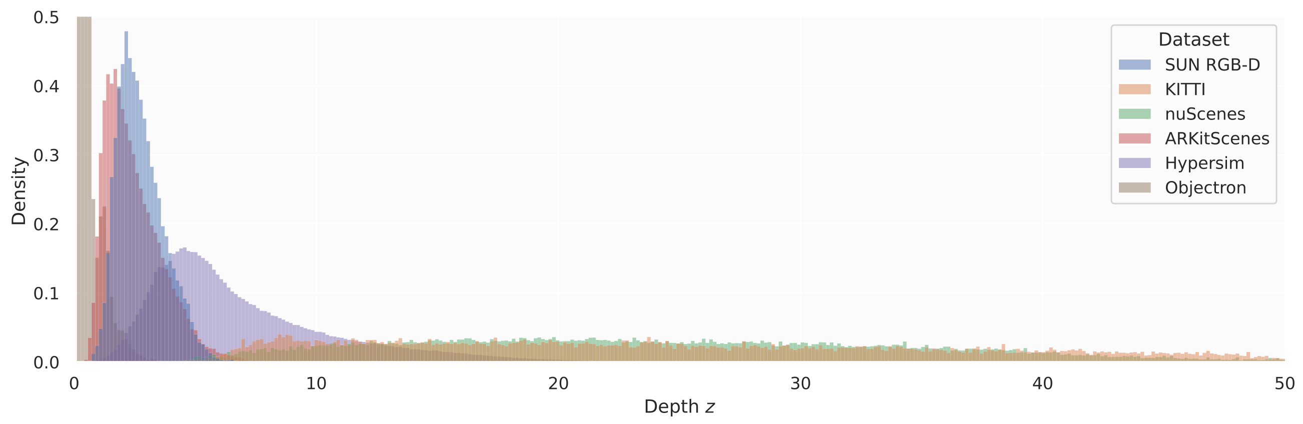


Figure 3. Normalized depth distribution per dataset in OMNI3D. We slightly limit the depth and density ranges for a better visualization.

2. Model Details

In this section, we provide more details for Cube R-CNN pertaining to its 3D bounding box allocentric rotation (Sec. 4.1) and the derivation of virtual depth (Sec. 4.2).

2.1. 3D Box Rotation

Our 3D bounding box object rotation is predicted in the form of a 6D continuous parameter, which is shown in [20] to be better suited for neural networks to regress compared to other forms of rotation. Let our predicted rotation $\mathbf{p}$ be split into two directional vectors $\mathbf{p_1}, \mathbf{p_2} \in \mathbb{R}^3$ and $\mathbf{r_1}, \mathbf{r_2}, \mathbf{r_3} \in \mathbb{R}^3$ be the columns of a 3×3 rotational matrix $\mathbf{R_a}$. Then $\mathbf{p}$ is mapped to $\mathbf{R_a}$ via

$$\mathbf{r_1} = \text{norm}(\mathbf{p_1}) \tag{1}$$

$$\mathbf{r_2} = \text{norm}(\mathbf{p_2} - (\mathbf{r_1} \cdot \mathbf{p_2})\mathbf{r_1}) \tag{2}$$

$$\mathbf{r_3} = \mathbf{r_1} \times \mathbf{r_2} \tag{3}$$

where $(\cdot, \times)$ denote dot and cross product respectively.

$\mathbf{R_a}$ is estimated in *allocentric* form similar to [10]. Let f_x, f_y, p_x, p_y be the known camera intrinsics, u, v the predicted 2D projected center as in Section 4.1, and $a = [0, 0, 1]$ be the camera's principal axis. Then $o = \text{norm}([\frac{u-p_x}{f_x}, \frac{v-p_y}{f_y}, 1])$ is a ray pointing from the camera to u, v with angle $\alpha = \text{acos}(o)$. Using standard practices of axis angle representations with an axis denoted as $o \times a$ and angle α, we compute a matrix $\mathbf{M} \in \mathbb{R}^{3\times3}$, which helps form the final *egocentric* rotation matrix $\mathbf{R} = \mathbf{M} \cdot \mathbf{R_a}$.

We provide examples of 3D bounding boxes at constant egocentric or allocentric rotations in Figure 4. The allocentric rotation is more aligned to the visual 2D evidence, whereas egocentric rotation entangles relative position into the prediction. In other words, identical egocentric rotations may look very different when viewed from varying spatial locations, which is *not* true for allocentric.

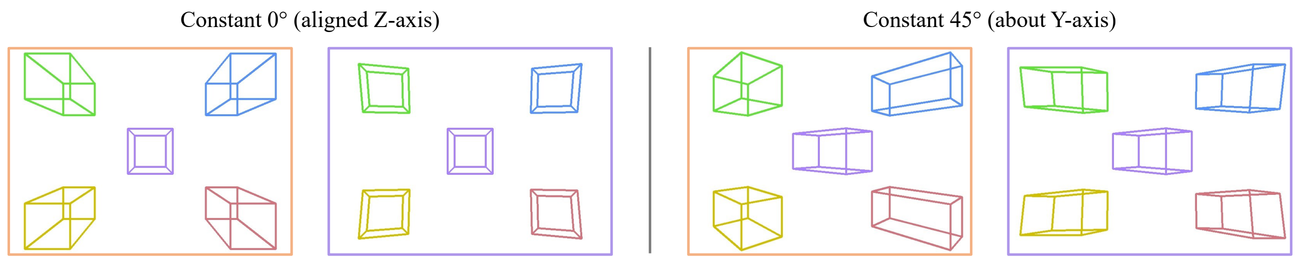


Figure 4. We show egocentric and allocentric representations under constant rotations. Despite being identical rotations, the egocentric representation appears visually different as its spatial location changes, unlike allocentric which is consistent with location changes.

2.2. Virtual Depth

In Section 4.2, we propose a virtual depth transformation in order to help Cube R-CNN handle varying input image resolutions and camera intrinsics. In our experiments, we show that virtual depth also helps other competing approaches, proving its effectiveness as a general purpose feature. The motivation of estimating a *virtual* depth instead of *metric* depth is to keep the effective image size and focal length consistent in an invariant camera space. Doing so enables two camera systems with nearly the same visual evidence of an object to transform into the same virtual depth as shown in Figure 4 of the main paper. Next we provide the proof for the conversion between virtual and metric depth.

Proof: Assume a 3D point (X, Y, Z) projected to (x, y) on an image with height H and focal length f. The virtual 3D point (X, Y, Z_v) is projected to (x_v, y_v) on the virtual image. The 2D points (x, y) and (x_v, y_v) correspond to the same pixel location in both the original and virtual image. In other words, $y_v = y \cdot \frac{H_v}{H}$. Recall the formula for projection as $y = f \cdot \frac{Y}{Z} + p_y$ and $y_v = f_v \cdot \frac{Y}{Z_v} + p_{y_v}$, where p_y is the principal point and $p_{y_v} = p_y \cdot \frac{H_v}{H}$. By substitution $f_v \cdot \frac{Y}{Z_v} = f \cdot \frac{Y}{Z} \frac{H_v}{H} \Rightarrow Z_v = Z \cdot \frac{f_v}{f} \frac{H}{H_v}$.

2.3. Training Details and Efficiency

When training on subsets smaller than OMNI3D, we adjust the learning rate, batch size, and number of iterations linearly until we can train for 128 epochs between 96k to 116k iterations. Cube R-CNN trains on V100 GPUs between 14 and 26 hours depending on the subset configuration when scaled to multi-node distributed training, and while training uses approximately 1.6 GB memory per image. *Inference* on a Cube R-CNN model processes image from KITTI [6] with a wide input resolution of 512×1696 at 52ms/image on average while taking up 1.3 GB memory on a Quadro GP100 GPU. Computed with an identical environment, our model efficiency is favorable to M3D-RPN [3] and GUP-Net [13] which infer from KITTI images at 191ms and 66ms on average, respectively.

3. Evaluation

In this section we give additional context and justification for our chosen thresholds τ which define the ranges of 3D IoU that AP_{3D} is averaged over. We further provide details on our implementation of 3D IoU.

3.1. Thresholds τ for AP_{3D}

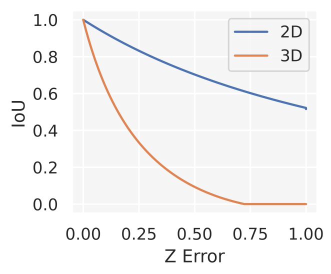
We highlight the relationship between 2D and 3D IoU in Figure 5. We do so by contriving a general example between a ground truth and a predicted box, both of which share rotation and rectangular cuboid dimensions $(0.5 \times 0.5 \times 1.0)$. We then translate the 3D box along the z-axis up to 1 meter (its unit length), simulating small to large errors in depth estimation. As shown in the left of Figure 5, the 3D IoU drops off significantly quicker than 2D IoU does between the projected 2D bounding boxes. As visualized in right of Figure 5, a moderate score of 0.63 IoU_{2D} may result in a low 0.01 IoU_{3D}. Despite visually appearing to be well localized in the front view, the top view helps reveal the error. Since depth is a key error mode for 3D, we find the relaxed settings of τ compared to 2D (Sec. 5) to be reasonable.

3.2. $\texttt{IoU}_{\text{3D}}$ Details

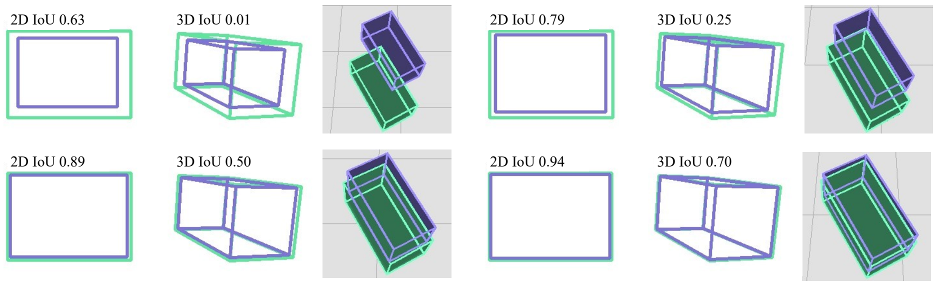
We implement a fast $\texttt{IoU}_{\text{3D}}$. We provide more details for our algorithm here. Our algorithm starts from the simple observation that the intersection of two oriented 3D boxes, b_1 and b_2, is a convex polyhedron with $n > 2$ comprised of connected planar units. In 3D, these planar units are 3D triangular faces. Critically, each planar unit belongs strictly to either b_1 or b_2. Our algorithm finds these units by iterating through the sides of each box, as described in Algorithm 1.

4. nuScenes Performance

In the main paper, we compare Cube R-CNN to competing methods with the popular IoU-based AP_{3D} metric, which is commonly used and suitable for general purpose 3D object detection. Here, we additionally compare our Cube R-CNN to best performing methods with the nuScenes evaluation metric, which is designed for the urban domain. The released nuScenes dataset has 6 cameras in total and reports custom designed metrics of mAP and NDS as de-



(a) IoUs vs Example Z Errors



(b) IoUs at Critical Points

Figure 5. We slowly translate a 3D box (purple) backwards relative to its ground-truth (green), hence simulating error in z up to 1 unit length ($\bar{l}_{3D}^{gt} = 1$). We plot 2D and 3D IoU vs z error in (a) and visualize selected critical points showing their 2D projections, 3D front view and a novel top view. Unsurprisingly, we find the drop in IoU_{3D} is much steeper than 2D. This effect highlights both the challenge of 3D object detection and helps justify the relaxed τ thresholds used for AP_{3D} in Section 5 of the main paper.

Algorithm 1: A high-level overview of our fast and exact $\texttt{IoU}_{3D}$ algorithm.

```
Data: Two 3D boxes $b_1$ and $b_2$
Result: Intersecting shape $S = []$
Step 1: For each 3D triangular face $e \in b_1$ we check
    wether $e$ falls inside $b_2$
Step 2: If $e$ is not inside, then we discard it
Step 3: If $e$ is inside, then $S = S + [e]$. If $e$ is
    partially inside, then the part of $e$ inside $b_2$, call it $\hat{e}$,
    is added to $S$, $S = S + [\hat{e}]$
Step 4: We repeat steps 1 - 3 for $b_2$
Step 5: We check and remove duplicates in $S$ (in the
    case of coplanar sides in $b_1$ and $b_2$)
Step 6: We compute the volume of $S$, which is
    guaranteed to be convex
```

tailed in [4]. Predictions from the 6 input views (each from a different camera) are fused to produce a single prediction. In contrast, our benchmark uses 1 camera (front) and therefore does not require or involve any post-processing fusion nor any dataset-specific predictions (e.g velocity and attributes). Moreover, as discussed in Section 5.1, the mAP score used in nuScenes is based on distances of 3D object centers which ignores errors in rotation and object dimensions. This is suited for urban domains as cars tend to vary less in size and orientation and is partially addressed in the NDS metric where true-positive (TP) metrics are factored in as an average over mAP and TP (Eq. 3 of [4]).

We compare with the current best performing methods on nuScenes, FCOS3D [19] and PGD [18]. We take their best

		nuScenes Front Camera Only					
Method	TTA	mAP	mATE	mASE	mAOE	NDS	AP_{3D}
FCOS3D [19]	✓	35.3	0.777	0.231	0.400	44.2	27.9
PGD [18]	✓	39.0	0.675	0.236	0.399	47.6	32.3
Cube R-CNN	✗	32.6	0.671	0.289	1.000	33.6	33.0

Table 2. We compare Cube R-CNN to competing methods, FCOS3D and PGD, on the nuScenes metrics on the single front camera setting, and without velocity or attribute computations factored into the NDS metric. The nuScenes metric uses a center-distance criteria at thresholds of $\{0.5, 1, 2, 4\}$ meters, ignoring object size and rotation, whereas our AP_{3D} metric (last col.) uses IoU_{3D}. Cube R-CNN predicts relative orientation to maximize IoU_{3D} rather than absolute, and therefore receives a high mAOE. Note that FCOS3D and PGD use test-time augmentations (TTA); we don't.

models including fine-tuning and test-time augmentations (TTA), then modify the nuScenes evaluation metric in the following three ways:

1. We evaluate on the single front camera setting.
2. We merge the construction vehicle and truck categories, as in the pre-processing of OMNI3D.
3. We drop the velocity and attribute true-positive metrics from being included in the NDS metric. Following [4] our TP = {mATE, mASE, mAOE} resulting in a metric of NDS $= \frac{1}{6}\left(3\text{ mAP} + \sum_{\text{mTP}\in\text{TP}} 1 - \min(1, \text{mTP})\right)$

Table 2 reports the performance of each nuScenes metric for Cube R-CNN, FCOS3D, and PGD along with our AP_{3D}. Since these methods are evaluated on the front cameras, no late-fusion of the full camera array is performed and side/back camera ground truths are not evaluated with. Cube R-CNN performs competitively but slightly worse on the mAP center-distance metric compared to FCOS3D and PGD. This is not surprising as the FCOS3D and PGD models were tuned for the nuScenes benchmark and metric and additionally use test time augmentations, while we don't.

Method	Trained on	table	bed	sofa	bathtub	sink	shelves	cabinet	fridge	chair	television	avg.
ImVoxelNet [16]	SUN RGB-D	39.5	68.8	48.9	33.9	18.7	2.4	13.2	17.0	55.5	8.4	30.6
Cube R-CNN	SUN RGB-D	39.2	65.7	58.1	49.0	32.5	4.3	16.2	25.2	54.5	2.7	34.7
Cube R-CNN	OMNI3D$_{\text{IN}}$	38.3	66.5	60.3	51.8	30.8	3.2	13.3	30.3	56.1	3.6	**35.4**

Table 3. Comparison to ImVoxelNet [16] on SUN RGB-D test. We use the full 3D object detection setting to report **AP$_{3D}$**.

Method	Trained on	table	bed	sofa	bathtub	sink	shelves	cabinet	fridge	chair	toilet	avg.
Total3D [14]	SUN RGB-D	27.7	33.6	30.1	28.5	18.8	10.1	13.1	19.1	24.2	28.1	23.3
Cube R-CNN	SUN RGB-D	39.2	49.5	46.0	32.2	31.9	16.2	26.5	34.7	39.9	45.7	36.2
Cube R-CNN	OMNI3D$_{\text{IN}}$	40.7	50.1	50.0	33.8	31.8	18.2	29.0	34.6	41.6	48.2	**37.8**

Table 4. Comparison to Total3D [14] on SUN RGB-D test. We use oracle 2D detections fairly for all methods and report **IoU$_{3D}$**.

5. Full Category Performance on OMNI3D

We train Cube R-CNN on the full 98 categories within OMNI3D, in contrast to the main paper which uses the 50 most frequent categories with more than $1k$ positive instances. As expected, the AP$_{3D}$ performance decreases when evaluating on the full categories from 23.3 to 14.1, and similarly for AP$_{2D}$ from 27.6 to 17.3. At the long-tails, we expect 2D and 3D object recognition will suffer since fewer positive examples are available for learning. We expect techniques related to few-shot recognition could be impactful and lend to a new avenue for exploration with OMNI3D.

6. Per-category SUN RGB-D Performance

We show per-category performance on AP$_{3D}$ for Cube R-CNN and ImVoxelNet [16]'s publicly released indoor model in Table 3. We show the 10 common categories which intersect all indoor datasets as used in Table 4-5 in the main paper for fair comparisons when training categories differ.

Similarly, Table 4 shows detailed per-category IoU$_{3D}$ performance for Cube R-CNN and Total3D [14] on 10 common categories, a summary of which was presented in Table 4 in the main paper. Note that *television*, in our 10 intersecting categories, is not detected by Total3D thus we replace it by *toilet* which is the next most common category.

7. Regarding the Public Models Used for OMNI3D Comparisons

We use publicly released code for M3D-RPN [3], GUP-Net [13], SMOKE [12], FCOS3D [19], PGD [18], and ImVoxelNet [16] and Total3D [14] in Section 5. Most of these models are implemented in the mmdetection3d [5] open-source repository, which critically supports many features for dataset scaling such as distributed scaling and strategic data sampling.

Most of the above methods tailor their configuration hyper-parameters in a handful of ways specifically for each dataset they were originally designed for. Since our experiments explicitly run on mixed datasets which have diverse resolutions, aspect ratios, depth distributions, *etc.*, we opted to run each method on a variety of settings. We did our best to run each method on multiple sensible hyper-parameters settings to give each method the most fair chance. We focused primarily on the settings which are impacted by input resolution and depth distributions, as these are seemingly the biggest distinctions between datasets. When applicable, we implemented virtual depth with a custom design for each method depending on its code structure (see main paper).

Although we expect that better recipes can be found for each method, **we ran more than 100 experiments** and are using 17 of these runs **selected based on best performance** to report in the paper.

8. Qualitative Examples

Figure 6 shows more Cube R-CNN predictions on OMNI3D test. In Figure 7, we demonstrate generalization for interesting scenes in the wild from COCO [11] images. When projecting on images with unknown camera intrinsics, as is the case for COCO, we visualize with intrinsics of $f = 2 \cdot H,\ p_x = \frac{1}{2}W,\ p_y = \frac{1}{2}H$, where $H \times W$ is the input image resolution. As shown in Figure 7, this appears to result in fairly stable generalization for indoor and more common failure cases concerning unseen object or camera poses in outdoor. We note that simple assumptions of intrinsics prevent our 3D localization predictions from being real-world up to a scaling factor. This could be resolved using either real intrinsics or partially handled via image-based self-calibration [8, 9, 21] which is itself a challenging problem in computer vision.

Lastly, we provide a demo video[1] which further demonstrates Cube R-CNN's generalization when trained on OMNI3D and then applied to in the wild video scenes from a headset mounted camera, similar to AR/VR heasets. In the video, we apply a simple tracking algorithm which merges predictions in 3D space using 3D IoU and category cosine similarity when comparing boxes. We show the image-based boxes on the left and a static view of the room on the right. We emphasize that this video demonstrates **zero-shot performance** since no fine-tuning was done on the domain data.

[1]https://omni3d.garrickbrazil.com/#demo

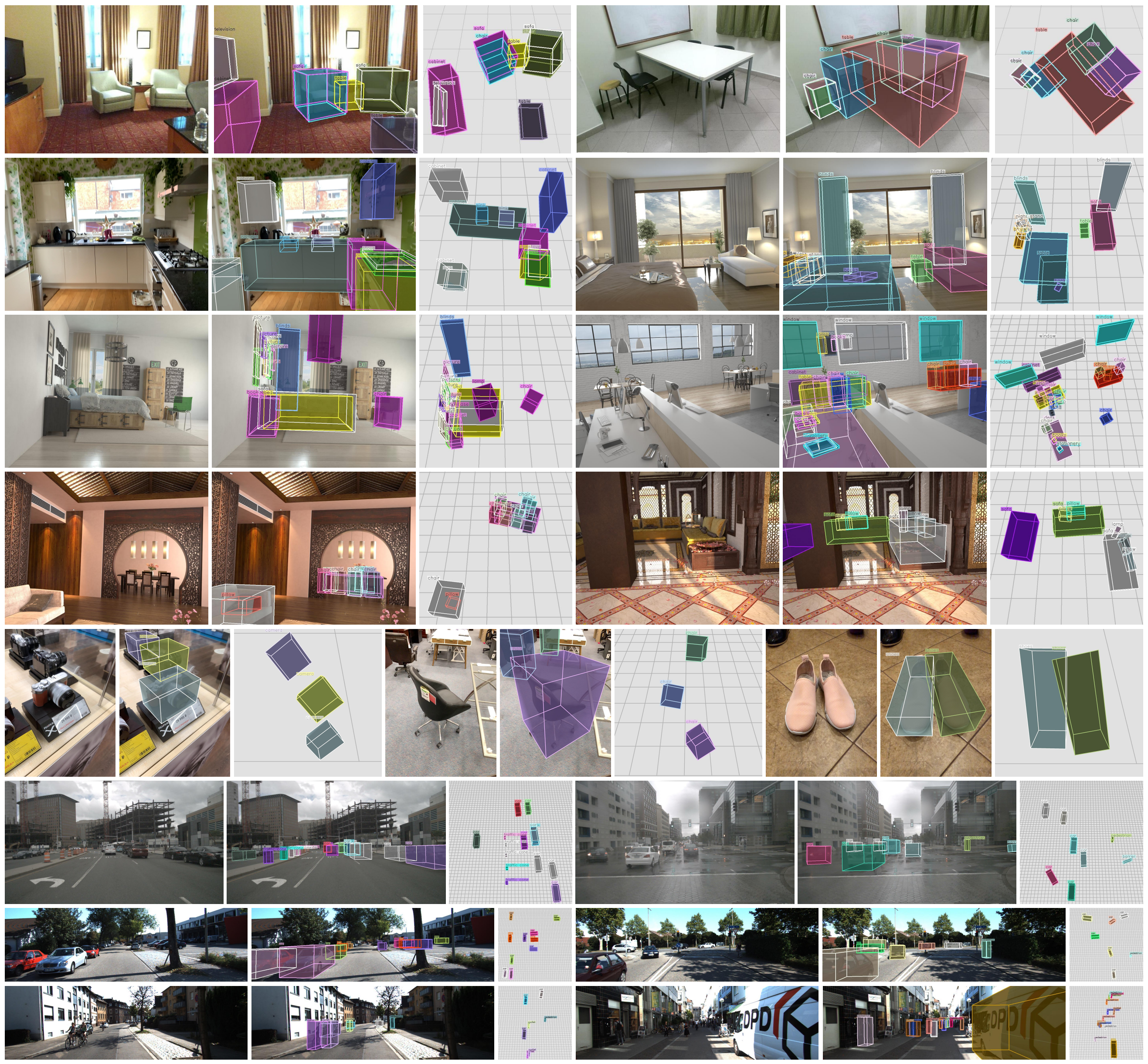


Figure 6. Cube R-CNN on OMNI3D test. We show the input image, the 3D predictions overlaid on the image and a top view. We show examples from SUN RGB-D [17], ARKitScenes [2], Hypersim [15], Objectron [1], nuScenes [4], and KITTI [6].

References

[1] Adel Ahmadyan, Liangkai Zhang, Artsiom Ablavatski, Jianing Wei, and Matthias Grundmann. Objectron: A large scale dataset of object-centric videos in the wild with pose annotations. *CVPR*, 2021. 1, 6

[2] Gilad Baruch, Zhuoyuan Chen, Afshin Dehghan, Tal Dimry, Yuri Feigin, Peter Fu, Thomas Gebauer, Brandon Joffe, Daniel Kurz, Arik Schwartz, and Elad Shulman. ARKitScenes - a diverse real-world dataset for 3D indoor scene understanding using mobile RGB-D data. In *NeurIPS Datasets and Benchmarks Track*, 2021. 1, 6

[3] Garrick Brazil and Xiaoming Liu. M3D-RPN: Monocular 3D region proposal network for object detection. In *ICCV*, 2019. 3, 5

[4] Holger Caesar, Varun Bankiti, Alex H Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom. nuScenes: A multimodal dataset for autonomous driving. In *CVPR*, 2020. 1, 4, 6

[5] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 5

[6] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the KITTI vision benchmark suite. In *CVPR*, 2012. 1, 3, 6

[7] Agrim Gupta, Piotr Dollar, and Ross Girshick. LVIS: A

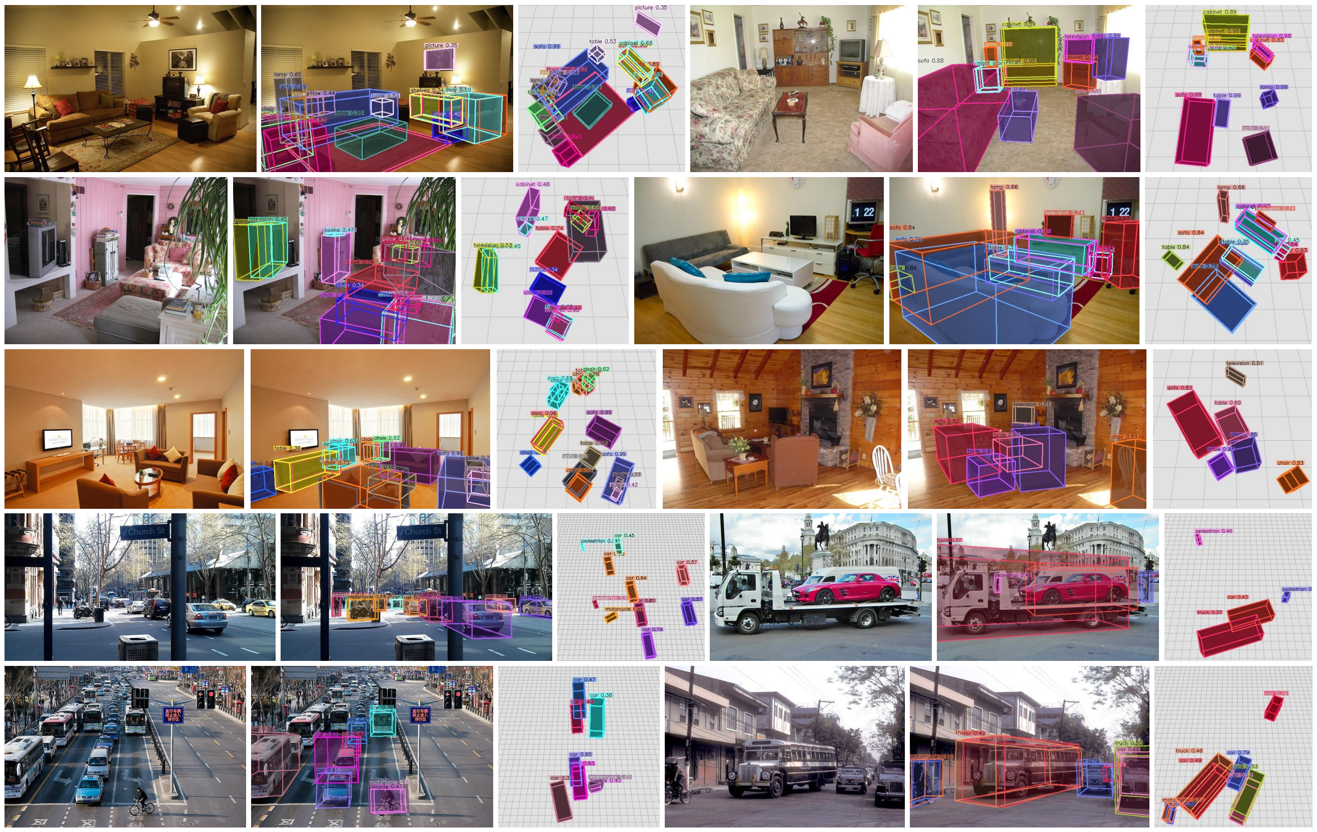


Figure 7. Cube R-CNN on COCO in the wild images. We select interesting scenes and observe that generalization performs well for indoor (Rows 1-3) compared to outdoor (Rows 4-5), which appear to fail when in the wild objects or cameras have high variation in unseen poses.

dataset for large vocabulary instance segmentation. In *CVPR*, 2019. 1

[8] Elsayed Hemayed. A survey of camera self-calibration. In *AVSS*, 2003. 5

[9] Razvan Itu and Radu Danescu. A self-calibrating probabilistic framework for 3D environment perception using monocular vision. *Sensors*, 2020. 5

[10] Abhijit Kundu, Yin Li, and James Rehg. 3D-RCNN: Instance-level 3D object reconstruction via render-and-compare. In *CVPR*, 2018. 2

[11] 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. 1, 5

[12] Zechen Liu, Zizhang Wu, and Roland Tóth. SMOKE: Single-stage monocular 3D object detection via keypoint estimation. In *CVPRW*, 2020. 5

[13] Yan Lu, Xinzhu Ma, Lei Yang, Tianzhu Zhang, Yating Liu, Qi Chu, Junjie Yan, and Wanli Ouyang. Geometry uncertainty projection network for monocular 3D object detection. In *ICCV*, 2021. 3, 5

[14] Yinyu Nie, Xiaoguang Han, Shihui Guo, Yujian Zheng, Jian Chang, and Jian Zhang. Total3DUnderstanding: Joint layout, object pose and mesh reconstruction for indoor scenes from a single image. In *CVPR*, 2020. 5

[15] Mike Roberts, Jason Ramapuram, Anurag Ranjan, Atulit Kumar, Miguel Bautista, Nathan Paczan, Russ Webb, and Joshua Susskind. Hypersim: A photorealistic synthetic dataset for holistic indoor scene understanding. In *ICCV*, 2021. 1, 6

[16] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. ImVoxelNet: Image to voxels projection for monocular and multi-view general-purpose 3D object detection. In *WACV*, 2022. 5

[17] Shuran Song, Samuel Lichtenberg, and Jianxiong Xiao. SUN RGB-D: A RGB-D scene understanding benchmark suite. In *CVPR*, 2015. 1, 6

[18] Tai Wang, Zhu Xinge, Jiangmiao Pang, and Dahua Lin. Probabilistic and geometric depth: Detecting objects in perspective. In *CoRL*, 2022. 4, 5

[19] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. FCOS3D: Fully convolutional one-stage monocular 3D object detection. In *ICCVW*, 2021. 4, 5

[20] Yi Zhou, Connelly Barnes, Jingwan Lu, Jimei Yang, and Hao Li. On the continuity of rotation representations in neural networks. In *CVPR*, 2019. 2

[21] Bingbing Zhuang, Quoc-Huy Tran, Gim Lee, Loong Fah Cheong, and Manmohan Chandraker. Degeneracy in self-calibration revisited and a deep learning solution for uncalibrated SLAM. In *IROS*, 2019. 5