EmbodiedScan: A Holistic Multi-Modal 3D Perception Suite Towards Embodied AI

Supplementary Materials

Tai Wang[1*], Xiaohan Mao[1,2*], Chenming Zhu[1,3*], Runsen Xu[1,4], Ruiyuan Lyu[1,5], Peisen Li[1,5], Xiao Chen[1,4], Wenwei Zhang[1], Kai Chen[1], Tianfan Xue[4], Xihui Liu[3], Cewu Lu[2], Dahua Lin[1,4], Jiangmiao Pang[1✉]

[1]Shanghai AI Laboratory [2]Shanghai Jiao Tong University [3]The University of Hong Kong
[4]The Chinese University of Hong Kong [5]Tsinghua University
[*]Equal contribution [✉]Corresponding author

https://github.com/OpenRobotLab/EmbodiedScan

1. Implementation Details

In the main paper, we focus on the overall design of our baseline framework and the benchmark results. Here, we further provide the implementation details of Embodied Perceptron and other baselines on our benchmark.

1.1. Embodied Perceptron

Input. Except for the monocular task, given the memory usage of different tasks, we set different numbers of input images during training and inference. Specifically, we set the number of input images to 20 and 50 for training and inference of multi-view 3D detection and visual grounding experiments while reducing the number to 10 for training continuous 3D detection models. For occupancy experiments, we set the number to 10 and 20 for training and inference. In addition, due to different resolutions of images from different source datasets, we resize them to 480×480 for unification and conduct corresponding transformations when computing the projection from points to images.

For depth maps, after converting them to point clouds, we first sample the points to limit their maximum number to 100k [14]. Then we voxelize them and feed them into the sparse convolutional networks. We set the voxel size to 0.01 meters for 3D detection following the convention of previous works [14]. In contrast, since we only use the last-level voxel feature (64× downsampled) to construct the feature volume, the voxel size is set to 0.16/64=0.0025 meters to ultimately predict the $40 \times 40 \times 16$ occupancy (the output voxel size is 0.16 meters). Finally, we summarize the settings of different benchmarks in Tab. 1.

Multi-Modal 3D Encoder. As mentioned in the main paper, we use the classical encoders for different modalities at the beginning, *i.e.*, a shared ResNet50 [6] for multi-view images, MinkResNet34 [4] for point clouds derived from depth maps, and RoBERTa-Base [7] for texts. Here, we reduce the base channels in ResNet to 16 to make it consistent with MinkResNet34, resulting in {128, 256, 512, 1024} multi-level feature channels after sparse fusion. In contrast, for dense fusion, we keep the original setting of ResNet, use FPN to enhance the 2D features (256 channels) to derive the 3D feature volume, and finally concatenate it with the densified last-level voxel feature V_4 (512 channels), resulting in 768-channel dense feature for subsequent occupancy prediction. For different outputs, the current encoder design has shared separated encoders but minor differences during fusion. How to further unify them and how it could benefit multi-task training and pre-training would be intriguing problems to be explored in the future.

Table 1. Benchmark inputs & outputs.

Benchmark	Inputs	Outputs
Cont. 3D Perception	Ordered 1∼ N Views	Visible 3D Boxes/Occ.
MV 3D Perception	Random N Views	All the 3D Boxes/Occ.
Mono. 3D Detection	1 Monocular View	Visible 3D Boxes
3D Visual Grounding	Random N Views & Prompt	Target 3D Object Box

Spare & Dense Decoder. We basically follow FCAF3D [14] in 3D detection head designs but adapt it to be compatible with oriented 3D boxes. It generates predictions based on sparse voxel seeds and assigns targets to them according to several rules during training, such as whether the voxel center is inside a box and assigning it to the best feature level similar to FCOS [18, 19]. Please see more details in its original paper. Here, all the computations regarding the distance between centers, points, and six faces and box formulations mentioned in the main paper are modified to fit the 3-DoF rotation version.

For the dense decoder, we first use a 3D FPN [15] to filter the 3D dense feature and compress the feature channel to 128. The output multi-level features in three resolutions, from $40 \times 40 \times 16$ to $10 \times 10 \times 8$, are fed into three occupancy prediction heads, which share the same architecture,

a 3D convolutional layer with kernel size and stride set to 1, to produce the multi-scale results. The training objective and loss are similar to SurroundOcc [20] for supervising the multi-scale output. During inference, we only take the high-resolution output as the final prediction.

For the visual grounding decoder, we adopt several transformer layers to fuse the 3D sparse feature and text feature. Similar to GroupFree3D [8], we refine the position encoding of an object candidate stage by stage. Specifically, we predict the 3D box locations at each decoder layer, and the predicted box location will be used to produce the updated position encoding of the same query. The queries are updated iteratively through $N_D = 6$ decoder layers. Besides, to achieve the contrastive loss mentioned in our original paper, one visual projection layer and one text projection layer are needed to project visual and text features to the same feature space with channel 64 for alignment. The projection layer consists of three linear layers. The contrastive loss aims to learn the similarity of visual-text multimodal features, consisting of two losses: $\mathcal{L}^v_{con}$ ensures the features of an object query are closer to positive text token features and farther from other text tokens, and $\mathcal{L}^t_{con}$ ensures the features of a target text token are closer to corresponding visual features and farther from other visual tokens.

$$\mathcal{L}^v_{con} = \sum_{i=1}^{k} -\log\left(\frac{\exp\left(\boldsymbol{o}_i^\top \boldsymbol{t}_i/\tau\right)}{\sum_{j=1}^{l}\exp\left(\boldsymbol{o}_i^\top \boldsymbol{t}_j/\tau\right)}\right), \tag{1}$$

$$\mathcal{L}^t_{con} = \sum_{i=1}^{l} -\log\left(\frac{\exp\left(\boldsymbol{t}_i^\top \boldsymbol{o}_i/\tau\right)}{\sum_{j=1}^{k}\exp\left(\boldsymbol{t}_i^\top \boldsymbol{o}_j/\tau\right)}\right), \tag{2}$$

$$\mathcal{L}_{con} = \mathcal{L}^v_{con} + \mathcal{L}^t_{con}, \tag{3}$$

where $\boldsymbol{o}$ and $\boldsymbol{t}$ are the object and text features after projection layers, and $\boldsymbol{o}^\top \boldsymbol{t}/\tau$ is their similarity. k and l are the number of objects and words. $\boldsymbol{t}_i$ is the positive word feature of the i-th candidate object.

Training Parameters. For all the experiments, we only use the pre-trained ResNet50 and RoBERTa provided by PyTorch while training other modules randomly initialized from scratch following end-to-end manners. The network is trained using AdamW [9] optimizer, with $\beta_1 = 0.9$, $\beta_2 = 0.999$. For continuous/multi-view/monocular 3D detection, we use 8 GPUs with 1/4/8 training samples on each to train the model for 96/120/24 epochs, setting the learning rate to 0.0002/0.001/0.0002 and weight decay to 0.0001. For all the occupancy experiments, we use 8 GPUs with 1 training sample on each to train the model for 24 epochs, setting the learning rate to 0.0001 and weight decay to 0.01.

Data Augmentation. Since the transformation of 3D boxes is easier than occupancy, we only conduct input data augmentations for 3D detection experiments. For continuous and multi-view 3D detection, we randomly flip and apply global transformations to the aggregated points, including random rotation with angles in $[-0.0873, 0.0873]$, random scaling with a ratio in $[0.9, 1.1]$ and random translation following a normal distribution with standard deviation 0.1, but do not apply any augmentation to images. The rotate and flip augmentations are removed for the view-dependent 3D visual grounding experiments. For monocular experiments, we use the same augmentation settings while also flipping 2D images for better performance.

1.2. 3D Detection Baselines

By default, our following re-implemented baselines use the same input setup (such as the number of input views) and backbone with Embodied Perceptron for consistent performance comparison, *e.g.*, ResNet50 for ImVoxelNet and FCOS3D, MinkResNet34 for FCAF3D, *etc.* All the baseline implementation starts with a basic adaptation for oriented 3D box prediction, a simple L1 loss on the Euler angles' cosine values, and can smoothly change the decoder formulation to ours for performance improvement, as mentioned in the main paper. We basically provide several key hyperparameters as follows and all the models are trained to fully converge. The re-implementation of baselines is based on their official release on top of MMDetection3D [5] and more details can be referred to our code release.

ImVoxelNet. We adapt its officially released code to fit our dataset and experiments. For multi-view 3D detection, we set the grid range to $[-3.2m \sim 3.2m, -3.2m \sim 3.2m, -0.78m \sim 1.78m]$ along the X-Y (horizontal) plane and Z (vertical) axis and adopt a random origin shift augmentation following a normal distribution with standard deviation $[0.7, 0.7, 0]$ along these axes as in the original paper. We use 8 GPUs with 1 training sample on each to train the model for 36 epochs, setting the learning rate to 0.0001 and weight decay to 0.0001. For monocular 3D detection, we change the range to $[-3.2m \sim 3.2m, -1.0m \sim 1.56m, 0.8m \sim 7.2m]$ along the XYZ axis where Y and Z correspond to the height and depth axis. We use 8 GPUs with 4 samples on each to train the model for 12 epochs, setting the learning rate to 0.0002 and the weight decay to 0.0001. All the backbones would use a $0.1\times$ smaller learning rate as in the original implementation.

VoteNet. We follow the official version for 3D detection on ScanNet and only change the orientation estimation to the trivial version mentioned above. We derive the mean 3D size of boxes according to our annotations for the partial bin-based box coder setup. We use 8 GPUs with 4/8 samples on each to train the model for 180/12 epochs for multi-view/monocular 3D detection and adopt data augmentations for point clouds including random flip, rotation, scaling, and translations, similar to our baseline. It was observed that the performance decreased a lot when changing the original multi-bin orientation estimation to the trivial one. It is also challenging for the original classification and localization design for our large-vocabulary setting. Therefore, how to

design a more effective head for such point-based methods is an important problem to be explored afterward.

ImVoteNet. Since adapting ImVoteNet for multi-view cases is non-trivial, we only implement it for monocular experiments for comparison. Following the official implementation, we first train a Faster R-CNN with the amodal 2D boxes derived from projected 3D boxes in the first stage and then tune the overall framework in the second stage. Related hyperparameters are set similar to VoteNet. We use 8 GPUs with 2/16 training samples on each to train the model for 8/24 epochs at the first/second stage, setting the learning rate to 0.02/0.001 and the weight decay to 0.0001/0.01 with the SGD/AdamW optimizer.

FCAF3D. FCAF3D is similar to our depth-only baseline, with differences in network designs such as multi-modality fusion and decoders. Thus, the hyperparameters are also set similarly to our baselines, including the input, optimizers, training epochs, and data augmentations.

FCOS3D. FCOS3D is a conventional baseline for monocular 3D detection with simple architecture designs. We follow the official implementation but change the backbone to ResNet50 for consistency. We use 8 GPUs with 8 training samples on each to train the model for 24 epochs, setting the learning rate to 0.024 and the weight decay to 0.0001 with the SGD optimizer.

1.3. Occupancy Prediction Baselines

We follow the original implementation of OccNet [16] and SurroundOcc [20] but change the BEV query according to our settings to finally derive the $40 \times 40 \times 16$ occupancy in the range $[-3.2m \sim 3.2m, -3.2m \sim 3.2m, -0.78m \sim 1.78m]$. Besides, we take the RGB-D sequence input in our setting as the multi-camera input in its original paper (for autonomous driving). The learning rate is set to 0.0002/0.0001 and the weight decay is set to 0.01/0.01 for the AdamW optimizer of OccNet and SurroundOcc. We use 8 GPUs with 1 sample on each to train the model for 48 epochs. Following the official code, we also use a $0.1\times$ smaller learning rate for their backbone weights update.

1.4. Visual Grounding Baselines

ScanRefer. We implement ReferNet [3] based on our adapted VoteNet and do not change other designs. We use 4 GPUs with 14 training samples on each to train the model for 48 epochs. The learning rate is set to be 1e-3 and the weight decay is set to 1e-5.

BUTD-DETR. We reimplement the BUTD-DETR in our codebase and also change the orientation estimation to achieve oriented 3D box prediction. Considering the large vocabulary setting in our benchmark, we do not predict the residual size of the 3D boxes based on their mean sizes calculated according to the annotations. Instead, following the original setting of BUTD-DETR, the essence of the visual grounding task is not to predict the 3D box and its category, but to predict the alignment score between one 3D box and the input prompt. We directly predict the actual size of each 3D box. Besides, we keep the input of the box stream unchanged as in its official implementation, *i.e.*, a pre-trained GroupFree3D detector is used to obtain 3D object box proposals, which are sent into the box stream. We use 4 GPUs with 24 training samples on each to train the model for 80 epochs. The learning rate of the backbone is set to be 1e-5, the text encoder is frozen and the learning rate of the remaining parts is set to be 1e-4.

L3Det. L3Det is a cleaner architecture that is modified based on BUTD-DETR, where the text and visual feature fusion is conducted in the decoder. Similar to BUTD-DETR, we change the orientation estimation to adapt to the oriented 3D box prediction, and other components remain unchanged. We modify BUTD-DETR and reimplement L3Det [23] as a cleaner architecture on top. The settings for training and optimization are the same as those of BUTD-DETR.

2. Dataset Details

2.1. Data Processing

Difference Among Source Datasets. In the main paper, we mentioned that although the source datasets all have RGB-D data, their data distributions have significant differences. Specifically, ScanNet provides the raw RGB-D sequence with the most frames (highest sampling frequency) and the image resolution 1296×968. 3RScan uses a portrait screen with image resolution 540×960 (but provides the image with 90° rotation, resulting in 960×540) and has fewer frames. Matterport3D directly provides general multi-view images with image resolution 1280×1024 instead of video sequences to serve as the image modality of 90 building-scale scenes. In the main paper, we have mentioned that we unify the input as a general multi-view case and sample ScanNet frames to make them consistent with the other two datasets. In addition, we rescale the images to 480×480 to extract 2D features to unify the resolutions of inputs and also force the 2D backbone to learn features robust to the scale and rotations.

2.2. Annotation

Definition of Oriented 3D Boxes. As mentioned in the main paper, we follow the typical definition of oriented 3D boxes, including 3D center, 3D size, and three Euler angles. This definition is naturally transferred from previous research in 3D detection, from 7-DoF boxes in autonomous driving to this 9-DoF version for any 3-DoF rotation. However, it still has some ambiguity in the definition of 3D sizes and orientations. Because the definition of length, width, and height is ambiguous, we define the 3D size as the length

along the XYZ axis, Δx, Δy, Δz, to constitute the 3D size. A potentially tricky problem is that we may have multiple solutions with different combinations of this 3D size and Euler angles for a specific oriented box. This is because we do not pre-define the 3D size for each object to just estimate the orientation, which is more similar to the settings of 6D pose estimation. As a result, our 9-DoF definition is more suitable for the detection setting considering that it should be more general for different objects from the large-vocabulary categories, but at the same time essentially can be reduced to a definition of boxes with eight vertices and a single normal/unidirectional orientation. It may lose other dimensions of orientation (compared to 6D pose) information, but in practice, such a unidirectional orientation is enough for most objects, considering many of them are symmetric. The discussion about such object representations and the corresponding evaluation metric design can be important in future works.

Language Prompt Generation. When producing language prompts, each prompt is designed to uniquely identify a target object within a 3D scan by establishing a distinct relationship between the target and an adjacent object, referred to as the "anchor". Following SR3D [1], we use the following compositional template to construct the language prompt:

$$\langle \text{target class} \rangle \ \langle \text{spatial relation} \rangle \ \langle \text{anchor class(es)} \rangle$$

We present the five types of spatial-relation language prompts to make this appendix self-contained: Horizontal Proximity, Vertical Proximity, Support, Allocentric, and Between [1]:

- **Horizontal Proximity:** The type of language prompt shows the distance in the horizontal direction between the target and anchor objects, which indicates how close or far the target is from the anchor in the scene.
- **Vertical Proximity:** This prompt indicates the vertical relationship between the target and anchor.
- **Between:** This prompt indicates there exists a target between the two anchors. Furthermore, we can obtain a more precise description, such as the target being in the middle of two anchors.
- **Allocentric:** Based on our new EmbodiedScan annotation, each object will contain precise *orientation* information. Based on the position vector between the target and anchor, as well as the orientation vector of the anchor object itself, we can easily determine whether the target is in front/back/left/right of the anchor.
- **Support:** This prompt indicates that the target is either supported by or supporting the anchor.

We generate each type of language prompt scene by scene. Before prompt generation, we need to separately filter out classes that are suitable as targets and anchors. In addition, for each scene, we need to further determine the valid class as the following:

- A class is a valid class for a target if: 1) There must be multiple objects of this class in the current scene. 2) The number of objects of this class in the current scene cannot exceed 6.
- A class is a valid class for an anchor if: 1) Objects of this class are unique in the current scene. 2) The number of objects of class in the current scene cannot exceed 6.
- Besides, an anchor can never belong to the same class as the target and, as such, its distractors.

Next, we elaborate on detailed rules for the generation of different spatial relationships:

- **Allocentric:** For each anchor, we traverse all the valid target classes. For all the objects of a certain target class, we calculate the positional vector between the target and anchor objects. Combined with the anchor's own orientation vector, we determine whether the target is in front of, behind, to the left, or to the right of the anchor. Note that an allocentric language prompt will only be generated when there is only one object belonging to a certain target class in a certain direction of this anchor.
- **Support and Vertical Proximity:** For each anchor and target object, we first calculate the Intersection over Union (IoU) of the anchor and target in the XY plane. If the IoU exceeds a certain threshold, we determine whether the anchor and target can form a support relationship based on a pre-defined list of supporter and supportee categories. The positional relationship, above or below, is judged based on the heights in the Z-axis direction.
- **Horizontal Proximity:** For each anchor, we traverse all the valid target classes. For all the objects of a certain target class, we calculate their distances to the anchor object. From these, we select the farthest and nearest objects to construct a language prompt for each.
- **Between:** Unlike other types of prompts, this prompt requires two anchors. We determine whether the target is between two anchors based on their top view 2D bounding boxes. Generally speaking, the target should be in the same Z range for each of the two anchors and be away from every other distractors by a certain distance.

2.3. Statistics

Complete Instance Statistics. We show the complete instance distribution in Fig. 1 for reference. It can be observed that it turns out a long-tailed distribution as expected and shows obvious superiority over previous datasets regarding the number of categories and instances.

Spatial Proximity Statistics. Here we first conduct a basic quantitative analysis of different categories of language prompts. We find that in the generated prompts, descriptions of Horizontal Proximity and Allocentric relationships accounted for the vast majority, while Vertical Proximity

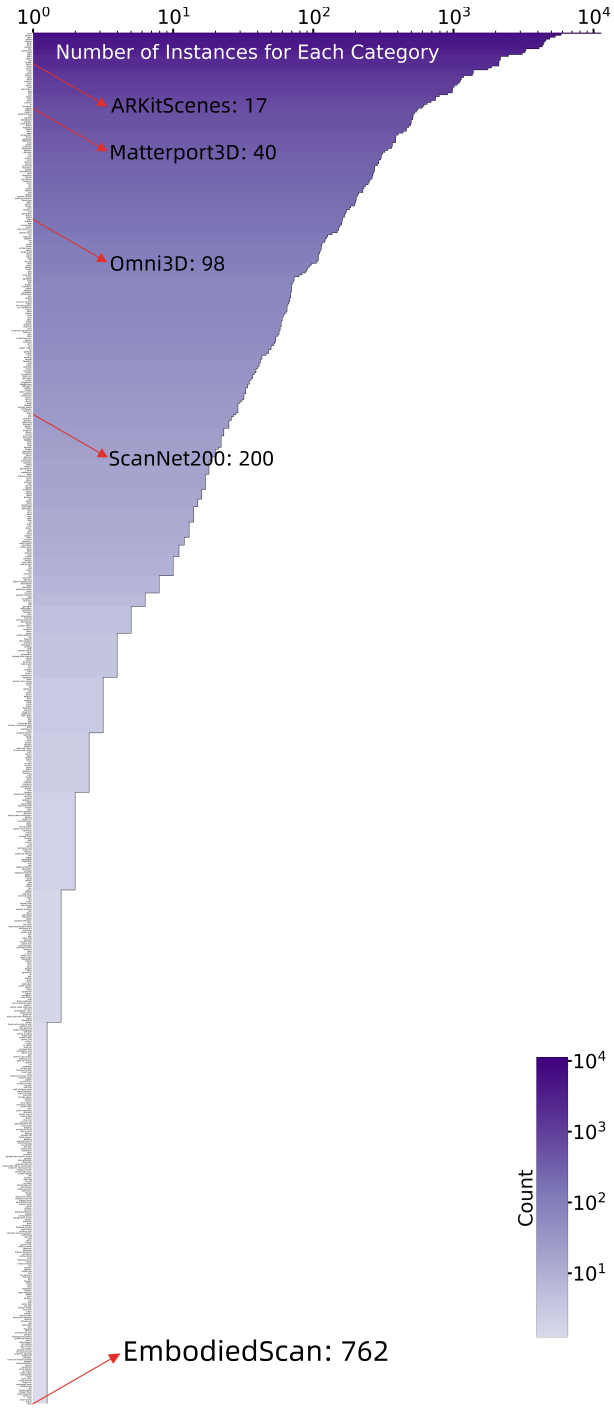


Figure 1. Complete instance distribution of EmbodiedScan.

Table 2. Spatial proximity statistics.

Horizontal	Vertical	Support	Allocentric	Between	All
723477	16420	4812	216197	9135	970041

and Support only made up a small portion. This is consistent with the fact that most objects in the 3D scene are primarily distributed on the XY plane (horizontal direction).

In addition, we make further analysis of the prompt distribution and discover several reasonable statistic results:

1. For common objects such as tables, we find that in the generated Support prompts, the ten most frequent objects are book, lamp, jacket, paper, plant, bottle, plate, box, telephone, and TV. By analyzing horizontal prompts, we find that nearby objects often include window, door, couch, cabinet, curtain, bin, and chair. This is actually consistent with our common sense, as tables are usually placed next to windows and accompanied by chairs for people to sit.

2. For smaller objects like books, we find that in the generated prompts, Support prompts account for the majority. Based on data analysis, books are usually placed on tables, stands, desks, cabinets, boxes, and dressers.

3. In the generated prompts, there are certain categories of objects usually appearing together within a scene. However, these objects are not common in our daily in-house lives, such as menu, cube, ridge, panel, sack, crate, and shovel.

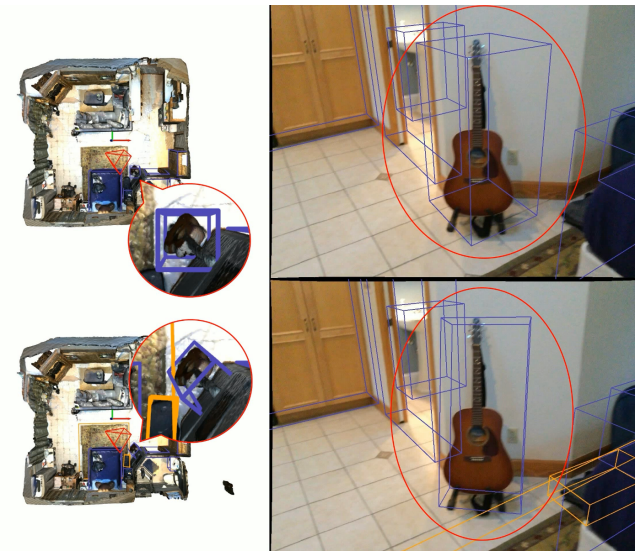
4. We also discovered several common spatial relationships. Some objects often appear in pairs. For example, mirrors frequently appear above the sink, stool, cabinet, and socket in the generated prompts. On the other hand, pictures are often placed above the bed, couch, table, desk, toilet, and cabinet.

These findings are small but interesting. We believe that given such detailed instance annotations, further analyses for the object distributions can reflect some common sense regarding the daily object configurations. It would also provide useful guidance for AI-powered realistic 3D scene generation and design.

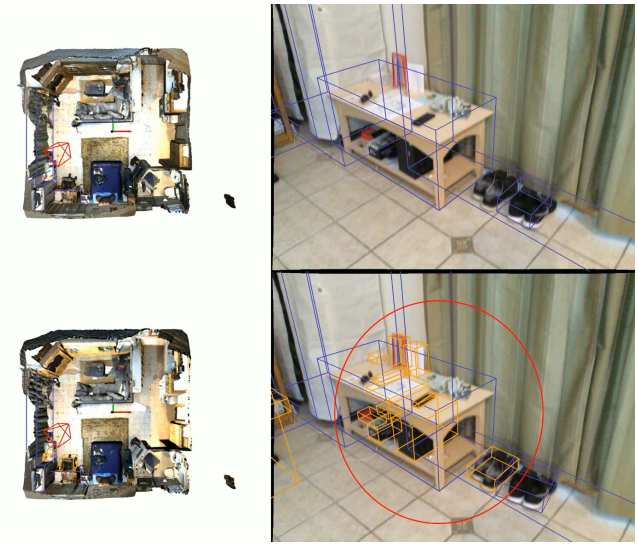
2.4. Data Examples

Oriented 3D Boxes. We show the comparison of previous and current annotations in Fig. 2. Here, we highlight the difference in box orientations and new annotated small objects in the provided two examples.

Complicated Language Prompts. Due to the fact that the language descriptions generated in the new benchmark contain more object categories, it is easier to generate prompts with ambiguity. Although we tried our best to ensure that the generated prompts have unique references through various restrictions during the generation procedure, this phenomenon still exists in our later manual inspection. Therefore, we randomly concatenate multiple generated language prompts belonging to the same object to generate complicated language prompts for auxiliary training. For example, for one object with three generated language prompts: "find the monitor that is closer to the door", "the monitor that is farthest from the windowsill" and "the monitor that is near the fan", we combine them to obtain the complicated language prompt: "find the monitor that is closer to the door, and it is farthest from the windowsill and near the fan."



(a) Box orientations.



(b) Small objects (new boxes marked in yellow).

Figure 2. Comparison of previous (top) and our annotations (bottom).

2.5. Clarifications

Dataset Comparison. First, we clarify several details in Tab. 1 of the main paper. For a fair and clear comparison, we modify some raw statistics of those datasets to make them consistent with the criterion of our dataset. For example, we do not show the number of objects for monocular datasets such as SUN RGB-D and Hypersim because these numbers can be inaccurate due to the potential repetitive counting of objects across different frames. We normalize the number of images from ScanNet by dividing its frame numbers by 10 to keep its sampling frequency consistent with ours. For categories, we only list the number of categories used for previous 3D detection instead of also involving that for 3D instance segmentation to highlight the much larger vocabulary of our 3D box annotations.

In addition, the number of language prompts shown in the Table is from ScanRefer [3]. There are also other annotations built upon ScanNet and we supplement these works here. ReferIt3D [1] is another work concurrent to ScanRefer but focuses on fine-grained 3D object identification, providing 120k prompts including spatial reference (SR3D) and natural reference (NR3D). ScanQA [2] targets the question-answering problem in 3D scenes and offers 41k question-answering pairs for ScanNet. SQA3D [10] further highlights the role of "situation" in this problem, resulting in 21k descriptions of 6.8k unique situations and 35k questions. All these works are built upon only ScanNet and thus have limited scene diversity. Recently, to collect large-scale 3D-text pairs for pre-training, 3D-VisTA [24] generates 278k scene descriptions from existing 3D Vision-Language tasks, templates, and GPT-3 from ScanNet and 3R-Scan as ScanScribe. It also randomly replaces objects from Objaverse with the same class to enhance the scene and object diversity, but at the same time, may yield a little domain gap between the generated and real-scanned raw data. In contrast, thanks to our comprehensive annotations for objects, regarding both categories and object poses, and more diverse scans, our preliminary version has a much larger scale in the language descriptions, scaling up the number to about 1M. Furthermore, due to the complicated scenes and object distributions, the task also becomes more challenging. We will continue to improve the existing language annotations and add more content from other aspects for holistic 3D scene understanding.

Test Set. We respect the copyright and license of all the source datasets and only include the test set statistics for scans and images in the main paper. For the annotations of the test set, we connect with the official hosts and will consider making a more complete version for future benchmarks and challenges. Other related issues will also be addressed by clear communication and collaboration with the official hosts.

3. Supplementary Results

Due to the space limitation, we only list the main benchmark results and key ablation studies to demonstrate the value of our dataset and the efficacy of our baseline. Here, we further show more details about these results, supplement more ablation studies, and visualize the predictions qualitatively both on our dataset and in the real world.

3.1. Detailed Benchmark Results

3D Detection Results Per Category. First, we show the detailed continuous and multi-view 3D detection performance

Table 3. Continuous and multi-view 3D detection results per category.

Methods	mAP_{25}	chair	picture	door	pillow	cabinet	table	book	window	box	shelf	plant	bin	curtain	bottle	lamp	couch	towel	sink
Camera-Only	12.80	72.39	1.30	26.63	25.07	21.08	55.03	2.81	6.16	8.25	25.37	28.20	42.66	8.84	0.01	12.87	75.90	0.46	26.93
Depth-Only	17.16	80.68	6.88	29.77	40.89	22.25	67.83	1.28	31.61	10.02	45.93	29.90	28.52	6.01	2.55	31.85	70.14	45.08	67.16
Multi-Modality	**19.07**	80.73	15.59	35.45	51.46	25.22	62.14	4.56	24.82	10.45	45.98	48.32	30.07	14.60	2.02	34.62	78.12	40.72	65.68
ImVoxelNet [15]	6.15	69.76	0.41	10.79	15.38	14.43	45.63	1.77	4.82	5.50	13.36	12.96	29.96	9.79	0.01	15.44	54.51	1.94	22.76
VoteNet [12]	3.20	64.92	0.00	0.01	3.30	3.24	30.42	0.12	0.05	0.91	0.02	1.84	17.17	0.39	0.00	5.55	33.37	0.11	10.88
FCAF3D [14]	9.07	86.98	2.42	9.01	44.54	21.03	54.20	15.02	10.71	7.13	24.65	23.22	56.93	17.86	0.47	27.11	63.56	11.80	63.74
+our decoder	14.80	90.30	17.01	42.82	49.22	36.01	67.20	20.94	30.26	9.83	41.93	30.40	70.51	39.44	1.13	35.33	76.99	36.01	72.58
+painting	15.10	90.79	20.25	45.70	52.30	36.98	67.42	18.55	31.23	11.30	40.98	33.14	70.28	38.27	0.91	34.50	73.70	30.45	73.43
Ours	**16.85**	88.81	19.57	42.36	54.65	38.78	67.12	20.59	33.69	12.92	40.97	35.48	71.18	43.85	1.52	37.36	77.65	31.74	72.92

Table 4. Monocular 3D detection results per category.

Methods	mAP_{25}	chair	pillow	cabinet	table	lamp	couch	desk	stand	bed	backpack
FCOS3D [19]	8.93	27.15	2.23	1.14	6.21	1.92	9.47	12.09	11.13	18.38	5.52
ImVoxelNet [15]	18.95	46.70	5.93	4.63	18.10	6.58	20.39	24.78	19.58	41.51	14.64
VoteNet [12]	14.30	54.00	1.65	2.41	19.53	3.55	21.80	19.13	4.89	45.58	4.21
ImVoteNet [13]	19.63	56.72	2.10	2.88	29.00	10.01	27.77	23.13	12.68	56.94	10.93
FCAF3D [14]	25.70	65.91	23.19	6.47	26.64	17.87	22.50	31.64	25.03	53.68	28.24
+our decoder	28.16	63.85	28.68	6.62	32.34	14.19	31.61	30.81	27.27	60.03	32.43
+painting	30.19	66.39	28.28	7.41	33.66	18.23	32.24	35.64	29.69	60.04	37.92
Ours	**34.28**	69.47	31.64	10.01	37.29	19.73	31.67	39.07	32.01	63.27	37.89

Methods	mAP_{25}	bathtub	ottoman	dresser	bin	toilet	refri.	stove	microwave	monitor	computer
FCOS3D [19]	8.93	6.31	1.38	4.54	10.23	40.51	6.92	4.03	5.60	3.25	0.57
ImVoxelNet [15]	18.95	10.14	9.63	9.98	17.82	65.70	18.11	15.92	14.93	8.26	5.80
VoteNet [12]	14.30	13.49	7.60	0.53	14.72	68.16	0.96	1.35	0.16	1.26	1.08
ImVoteNet [13]	19.63	37.56	9.14	1.87	21.96	74.08	1.21	9.50	2.12	2.24	0.66
FCAF3D [14]	25.70	26.38	15.76	4.35	34.93	71.90	13.88	4.29	9.95	21.57	9.79
+our decoder	28.16	38.17	21.85	7.28	38.96	75.57	16.25	7.78	10.31	6.13	13.04
+painting	30.19	41.31	20.23	7.16	42.86	77.59	16.12	9.56	10.76	14.04	14.68
Ours	**34.28**	50.63	25.59	9.54	45.17	80.39	24.44	14.53	19.96	19.77	23.65

in Tab. 3 for categories that are common in the real world and annotations. We can see that although these categories have a large number of annotations, there are still some that seem challenging for current models, such as pictures and bottles. In addition, it can be observed that the improvement brought by a better decoder for orientation estimation (+our decoder) is mainly focused on those objects that have significant differences between length and width, such as pictures, doors, windows, shelves, towels, *etc.* It is a reasonable phenomenon and reveals the importance of orientation estimation in this setting. Finally, because we do not list the 20 categories in the monocular 3D detection benchmark, we show the complete results in Tab. 4.

Summarized Key Takeaways. We mainly discuss our dataset's value and provide detailed benchmark results with respective discussions for each benchmark in the main paper. Here, we summarize other common key takeaways of those results as follows:

1. RGB-D outperforms RGB/depth-only as expected and the role of RGB is more crucial in occupancy prediction.

2. Our settings pose new challenges, particularly in the large vocabulary and orientation estimation for conventional tasks, leading to some occasional results such as VoteNet can perform a little worse than ImVoxelNet.

3. Monocular cases are notably difficult without stereo cues and with truncated object views. Grounding is more challenging due to complex prompts that avoid ambiguity.

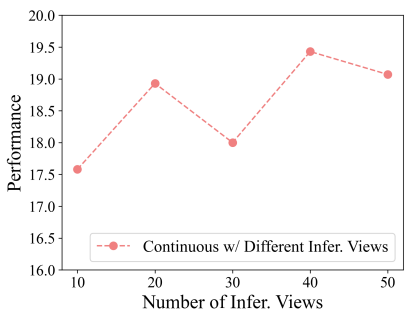
Table 5. We implement more methods on multi-view 3D detection given the similarity to their papers' settings. Monocular experiments and other methods need to adapt more operations tailored to 9-DoF boxes and will be added afterward, *e.g.*, 3DETR needs a 9-DoF GIoU loss and more innovations to handle large vocabulary well similar to 2D DETR methods. We implement a simplified/approximated version, achieving 62.87% AP on "bed" but still very low performance on most categories. ($\sim$1% mAP overall).

Method	mAP_{25}	mAR_{25}	$Head_{25}$	$Common_{25}$	$Tail_{25}$
H3DNet [22]	3.72	6.23	7.55	1.97	1.07
NeRF-Det [21]	7.90	22.91	13.12	5.74	3.35
Ours	**16.85**	51.07	28.65	12.83	7.09

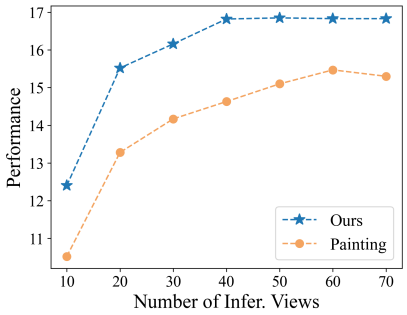
3.2. Supplementary Main Results and Discussions

More Recent Baselines. The main paper mainly implements classical baselines from different streams (image/point/voxel-based) for each benchmark. We include more recent ones in Tab. 5. For language-grounded benchmarks, given the difficulty of zero-shot methods handling our settings, we do not compare with them for a fair comparison. Given OpenScene's [11] inability to distinguish instances, here we adapt OpenMask3D [17] on our benchmark by replacing instance masks with our 9-DoF box proposals, resulting in 6.64% vs. our 25.72% AP_{25}.

Compared with zero-shot methods using CLIP. It is a natural idea to incorporate CLIP in our framework as other zero-shot multi-modal 3D perception methods. We are also considering it to enhance multi-modal features. However, it



(a) Continuous 3D detection with different numbers of inference views.



(b) Multi-view 3D detection with different numbers of inference views.

(c) Multi-view 3D detection with different numbers of training views.

Figure 3. Performance changed with the number of training and inference views.

Table 6. Ablation studies for dense fusion.

Methods	Input	mIOU	empty	floor	wall	chair	cabinet	door	table	couch	shelf	window	bed	curtain	refri.	plant	stairs	toilet
Painting	RGB-D	20.33	77.45	70.49	57.58	55.11	31.51	22.29	55.61	47.75	40.27	28.16	50.32	41.52	19.14	19.66	13.64	45.43
MinkUNet	RGB-D	24.53	71.10	63.41	56.19	49.99	35.05	37.16	50.96	46.87	38.30	33.56	54.97	41.49	30.20	35.89	12.77	64.56
MinkResNet (w/o FPN)	RGB-D	21.16	77.45	70.81	57.09	55.44	30.99	23.49	55.31	49.64	40.59	30.65	48.58	38.96	19.10	22.37	6.77	58.57
MinkResNet (Ours)	RGB-D	**27.65**	77.57	71.04	62.12	57.30	38.31	41.09	56.79	50.72	46.06	38.75	56.24	46.38	29.47	40.52	17.44	68.95

Table 7. More experiments with real and rendered images.

Train	Val	mAP_{25}	$Head_{25}$	$Common_{25}$	$Tail_{25}$
Render	Render	22.11	33.01	16.44	6.74
Render	Real	18.72	27.02	14.85	6.25
Real	Real	21.98	32.91	17.18	5.05
Real	Render	22.13	32.34	17.81	6.00
Real+Render	Real	22.26	33.48	16.99	5.46
Real+Render	Render	22.87	34.56	17.29	5.51

can only enable limited *simple* tasks like open-vocabulary semantic segmentation. For more *difficult* scenarios like *3D geometric understanding and grounding across different views* in our benchmark, additional data is needed. Our dataset, along with feature fusion and decoders, becomes crucial for fine-tuning such pre-trained models with 3D backbones.

More Experiments of Using Real/Rendered Data. It's nice to see our model trained with real captures also generalizes well to rendered views (Tab. 7). Combining these two data sources further brings limited improvement.

3.3. Supplementary Ablation Studies

Number of Views. As mentioned in the main paper, our trained baseline is applicable to any number of views during training and inference. Here, we show the ablation study regarding the number of views used for training and inference in Fig. 3. Specifically, taking 3D detection as the example, we change the inference views for continuous and multi-view settings and record the performance change in Fig. 3a and 3b. We can see that it has a relatively minor influence on the continuous setting because the ground truth also changes as the visible instances become fewer when reducing the number of inference views. For multi-view experiments, it affects the performance only when the number of views is too small (*e.g.*, < 20), but it is more robust than the simple painting baseline, potentially benefiting from the stronger multi-modality fusion. Finally, similar to inference, it is also better to use more views for training but would saturate when using more than 20 views (Fig. 3c). Therefore, setting the number of training views to 20 is a good trade-off between training costs and performance.

Dense Fusion. Here, we provide more comparison results with other RGB-D baselines for dense occupancy prediction. Taking the multi-view occupancy prediction as an example, we re-implement the painting baseline as the detection experiments and observe a much lower performance, as shown in Tab. 6. We conjecture it is because the painting loses much more dense information with such sparse feature extraction, resulting in such a larger gap from our dense fusion method. As for the alternative choices for dense fusion, we first attempt to voxelize the space with 0.16m voxels and use a MinkUNet to produce the sparse voxel feature for subsequent fusion and dense prediction. It turns out that fine-grained partition is necessary at the beginning. Besides, if removing the FPN to make the 2D feature extraction more lightweight, we cannot obtain the final competitive performance either.

Sparse Fusion. However, the FPN is not necessary for sparse fusion, especially considering the optimization problem encountered in the 3D detection baseline. Furthermore, except for the unstable training problem mentioned in the main paper, our baseline is also much more computationally efficient than the alternative implementations, which keep the FPN or paint the points with image features. Our final baseline costs only $\sim$25G of memory with the reduction of 2D feature channels and removing the FPN, compared to $\sim$59G of memory used in other approaches. Fi-

Table 8. Ablation studies for sparse fusion designs. xyz+2D feat paints input points with sampled 2D features, and 3D+2D feat uses multi-level voxels but only the last 2D feature layer for sampling.

Method	mAP_{25}	$Head_{25}$	$Common_{25}$	$Tail_{25}$
xyz+rgb (painting)	20.78	30.69	15.41	7.15
xyz+2D feat	20.73	31.38	15.35	5.41
3D+2D feat	21.06	32.09	15.10	5.84
ML-3D+2D feat (ours)	**21.70**	31.77	16.89	6.77

Table 9. Ablation studies of designs for sparse decoder.

Method	mAP_{25}	mAP_{50}	$Head_{25}$	$Common_{25}$	$Tail_{25}$
w/o Decouple	20.14	11.89	30.71	14.06	6.16
Decouple (sum.)	18.03	9.98	27.16	12.85	5.86
Decouple (avg.)	21.50	11.30	32.13	17.62	3.71
+Norm by 3D Size	20.67	11.39	30.68	16.20	5.28
Decouple (weigh.)	**21.70**	12.53	31.77	16.89	6.77
7-DoF IoU Loss	21.51	14.43	32.21	16.03	6.22
+ Corner Loss	**22.13**	13.95	32.60	16.82	7.08

nally, we provide the ablation study for alternative sparse fusion methods in Tab. 8. Our method, adopting multi-level feature fusion, empirically outperforms others, thanks to consistent inherent 2D/3D feature space and gradient back-propagation.

Sparse Decoder. The basic comparison between the simple L1 loss for Euler angles and our final decoder has been shown in the main paper. Here, we elaborate more on the ablation results of several design details in Tab. 9. We first compare the results of different combination methods for the corner losses and see that the weighted disentangled loss shows the best performance, compared to "w/o decouple", "simple summation", and "average". In addition, taking the size of boxes into consideration and normalizing the corner losses by their sizes cannot bring improvement in the final performance.

Apart from the corner loss, another straightforward approach is to implement a pseudo-3D-IoU loss for these methods. Specifically, since the computation of 3D IoU among 9-DoF boxes is still heavy and non-differentiable, we approximate the 3D boxes as 7-DoF ones with only the yaw part in the axis-aligned coordinate system. This "hack" method shows outstanding performance, especially in mAP_{50}, and can be further enhanced by combining the corner loss. Therefore, IoU-based loss is a design more faithful to the final metric and worthy of further study for the general 9-DoF case.

Performance with Different Training Data. During the procedure of scaling up data and annotations, we also test the model's performance on ScanNet and our final validation set. We can find a performance improvement that seems to be linear with respect to the number of scans (1.5k-3k-5k scans from ScanNet to EmbodiedScan), especially for objects with plenty of annotations ("Head" categories). We would continue collecting the RGB-D scan data and annotations to further push the model's performance to a higher level, towards the usage in practice and real-world embodied AI.

Table 10. Performance with different training data.

Train	Val	Overall	Head	Common	Tail
ScanNet	ScanNet	20.28	29.81	15.57	6.40
+3RScan	ScanNet	21.41	31.61	17.07	5.35
+Matterport3D	ScanNet	**23.02**	33.82	18.09	6.57
ScanNet	EmbodiedScan	10.92	21.10	8.06	1.78
+3RScan	EmbodiedScan	13.91	25.25	10.69	3.76
+Matterport3D	EmbodiedScan	**16.85**	28.65	12.83	7.09

3.4. Qualitative Results

We visualize the prediction results on EmbodiedScan in Fig. 4. From top to bottom, we plot the predictions of continuous 3D detection and occupancy prediction, monocular 3D detection, and multi-view 3D visual grounding. From this visualization, we can have a feeling about different perception output formats and how our models perform on our dataset. We can observe that the continuous perception can keep most previous predictions and fix some of them with the exploration. In addition, the localization of 3D visual grounding can be more accurate than the classical 3D detection for the target object considering it only needs to predict a single bounding box.

3.5. In-the-Wild Test Demo

Finally, we test our trained model in the wild. Specifically, we use Azure Kinect DK to record the RGB-D streams with camera poses and feed them into our models. We only modify the IoU/score thresholds to 0.15/0.075 and visualize $\leq$10 objects per category to avoid redundant boxes. It shows decent performance in our three test cases without cherry-picking, even with a different RGB-D sensor in different environments potentially having significant domain gaps from the training data. We visualize the prediction results in the attached supplementary video.

4. Supplementary Video

To provide a summary of our paper and key contributions, we make a short supplementary video, starting from the in-the-wild test demo to the underlying dataset, annotation tools, and methodology, and attach it to the supplementary materials.

References

[1] Panos Achlioptas, Ahmed Abdelreheem, Fei Xia, Mohamed Elhoseiny, and Leonidas Guibas. Referit3d: Neural listeners for fine-grained 3d object identification in real-world scenes. In *European conference on computer vision*, 2020. 4, 6

[2] Daichi Azuma, Taiki Miyanishi, Shuhei Kurita, and Motoaki Kawanabe. Scanqa: 3d question answering for spatial scene understanding. In *proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022. 6

[3] Dave Zhenyu Chen, Angel X Chang, and Matthias Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. In *European conference on computer vision*, 2020. 3, 6

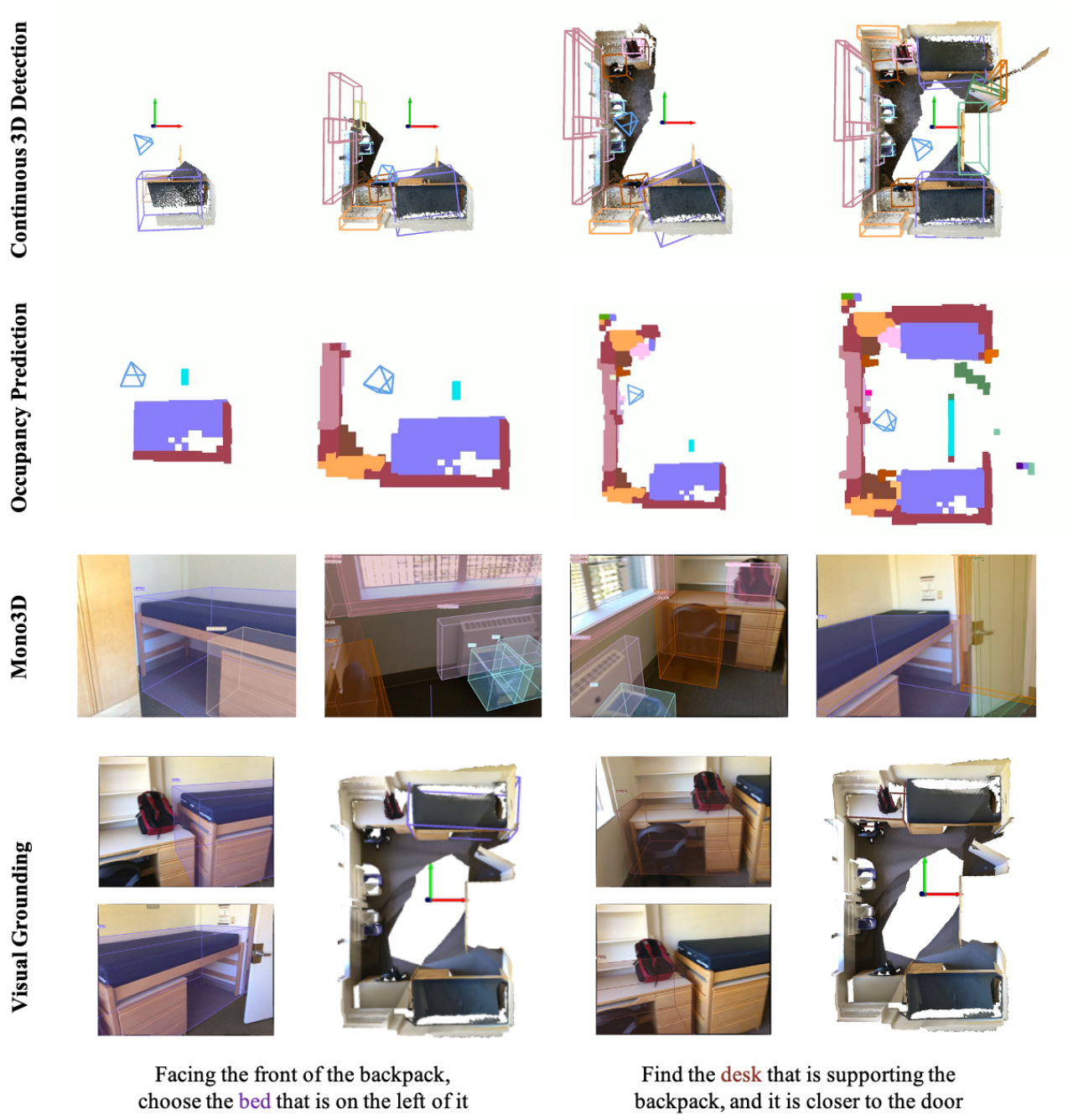


Figure 4. Qualitative results of different tasks on EmbodiedScan.

[4] Christopher Choy, JunYoung Gwak, and Silvio Savarese. 4d spatio-temporal convnets: Minkowski convolutional neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[5] MMDetection3D Contributors. MMDetection3D: OpenMMLab next-generation platform for general 3D object detection. https://github.com/open-mmlab/mmdetection3d, 2020. 2

[6] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2016. 1

[7] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettle-

moyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019. 1

[8] Ze Liu, Zheng Zhang, Yue Cao, Han Hu, and Xin Tong. Group-free 3d object detection via transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021. 2

[9] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. 2

[10] Xiaojian Ma, Silong Yong, Zilong Zheng, Qing Li, Yitao Liang, Song-Chun Zhu, and Siyuan Huang. Sqa3d: Situated question answering in 3d scenes. *arXiv preprint arXiv:2210.07474*, 2022. 6

[11] Songyou Peng, Kyle Genova, Chiyu Jiang, Andrea Tagliasacchi, Marc Pollefeys, Thomas Funkhouser, et al. Openscene: 3d scene understanding with open vocabularies. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023. 7

[12] Charles R Qi, Or Litany, Kaiming He, and Leonidas J Guibas. Deep hough voting for 3d object detection in point clouds. In *proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019. 7

[13] Charles R Qi, Xinlei Chen, Or Litany, and Leonidas J Guibas. Imvotenet: Boosting 3d object detection in point clouds with image votes. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020. 7

[14] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. Fcaf3d: Fully convolutional anchor-free 3d object detection. In *European Conference on Computer Vision*, 2022. 1, 7

[15] Danila Rukhovich, Anna Vorontsova, and Anton Konushin. Imvoxelnet: Image to voxels projection for monocular and multi-view general-purpose 3d object detection. In *WACV*, pages 2397–2406, 2022. 1, 7

[16] Chonghao Sima, Wenwen Tong, Tai Wang, Li Chen, Silei Wu, Hanming Deng, Yi Gu, Lewei Lu, Ping Luo, Dahua Lin, and Hongyang Li. Scene as occupancy. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 3

[17] Ayça Takmaz, Elisabetta Fedele, Robert W Sumner, Marc Pollefeys, Federico Tombari, and Francis Engelmann. Openmask3d: Open-vocabulary 3d instance segmentation. *arXiv preprint arXiv:2306.13631*, 2023. 7

[18] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *IEEE Conference on Computer Vision and Pattern Recognition*, 2019. 1

[19] Tai Wang, Xinge Zhu, Jiangmiao Pang, and Dahua Lin. FCOS3D: Fully convolutional one-stage monocular 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 2021. 1, 7

[20] Yi Wei, Linqing Zhao, Wenzhao Zheng, Zheng Zhu, Jie Zhou, and Jiwen Lu. Surroundocc: Multi-camera 3d occupancy prediction for autonomous driving. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 2, 3

[21] Chenfeng Xu, Bichen Wu, Ji Hou, Sam Tsai, Ruilong Li, Jialiang Wang, Wei Zhan, Zijian He, Peter Vajda, Kurt Keutzer, et al. Nerf-det: Learning geometry-aware volumetric representation for multi-view 3d object detection. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 23320–23330, 2023. 7

[22] Zaiwei Zhang, Bo Sun, Haitao Yang, and Qixing Huang. H3dnet: 3d object detection using hybrid geometric primitives. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XII 16*, pages 311–329. Springer, 2020. 7

[23] Chenming Zhu, Wenwei Zhang, Tai Wang, Xihui Liu, and Kai Chen. Object2scene: Putting objects in context for open-vocabulary 3d detection. *arXiv preprint arXiv:2309.09456*, 2023. 3

[24] Ziyu Zhu, Xiaojian Ma, Yixin Chen, Zhidong Deng, Siyuan Huang, and Qing Li. 3d-vista: Pre-trained transformer for 3d vision and text alignment. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023. 6