
3D-LLM: Injecting the 3D World into Large Language Models

Yining Hong

University of California, Los Angeles

Haoyu Zhen

Shanghai Jiao Tong University

Peihao Chen

South China University of Technology

Shuhong Zheng

University of Illinois Urbana-Champaign

Yilun Du

Massachusetts Institute of Technology

Zhenfang Chen

MIT-IBM Watson AI Lab

Chuang Gan

UMass Amherst and MIT-IBM Watson AI Lab

Abstract

Large language models (LLMs) and Vision-Language Models (VLMs) have been proven to excel at multiple tasks, such as commonsense reasoning. Powerful as these models can be, they are not grounded in the 3D physical world, which involves richer concepts such as spatial relationships, affordances, physics, layout, and so on. In this work, we propose to inject the 3D world into large language models and introduce a whole new family of 3D-LLMs. Specifically, 3D-LLMs can take 3D point clouds and their features as input and perform a diverse set of 3D-related tasks, including captioning, dense captioning, 3D question answering, task decomposition, 3D grounding, 3D-assisted dialog, navigation, and so on. Using three types of prompting mechanisms that we design, we are able to collect over 300k 3D-language data covering these tasks. To efficiently train 3D-LLMs, we first utilize a 3D feature extractor that obtains 3D features from rendered multi-view images. Then, we use 2D VLMs as our backbones to train our 3D-LLMs. By introducing a 3D localization mechanism, 3D-LLMs can better capture 3D spatial information. Experiments on ScanQA show that our model outperforms state-of-the-art baselines by a large margin (*e.g.*, the BLEU-1 score surpasses state-of-the-art score by 9%). Furthermore, experiments on our held-in datasets for 3D captioning, task composition, and 3D-assisted dialogue show that our model outperforms 2D VLMs. Qualitative examples also show that our model could perform more tasks beyond the scope of existing LLMs and VLMs. Project Page: : <https://vis-www.cs.umass.edu/3d11m/>.

1 Introduction

In the past several years, we have witnessed a surge of large language models (LLMs) (*e.g.*, GPT4 [33]) that excel at multiple tasks, such as communication and commonsense reasoning. Recent works have explored aligning images and videos with LLM for a new generation of multi-modal LLMs (*e.g.*, Flamingo [14], BLIP-2 [30]) that equip LLMs with the ability to understand and reason about 2D images. However, as powerful as the models can be in communication and reasoning, they are not *grounded in the real 3D physical world*, which involves richer concepts such as spatial

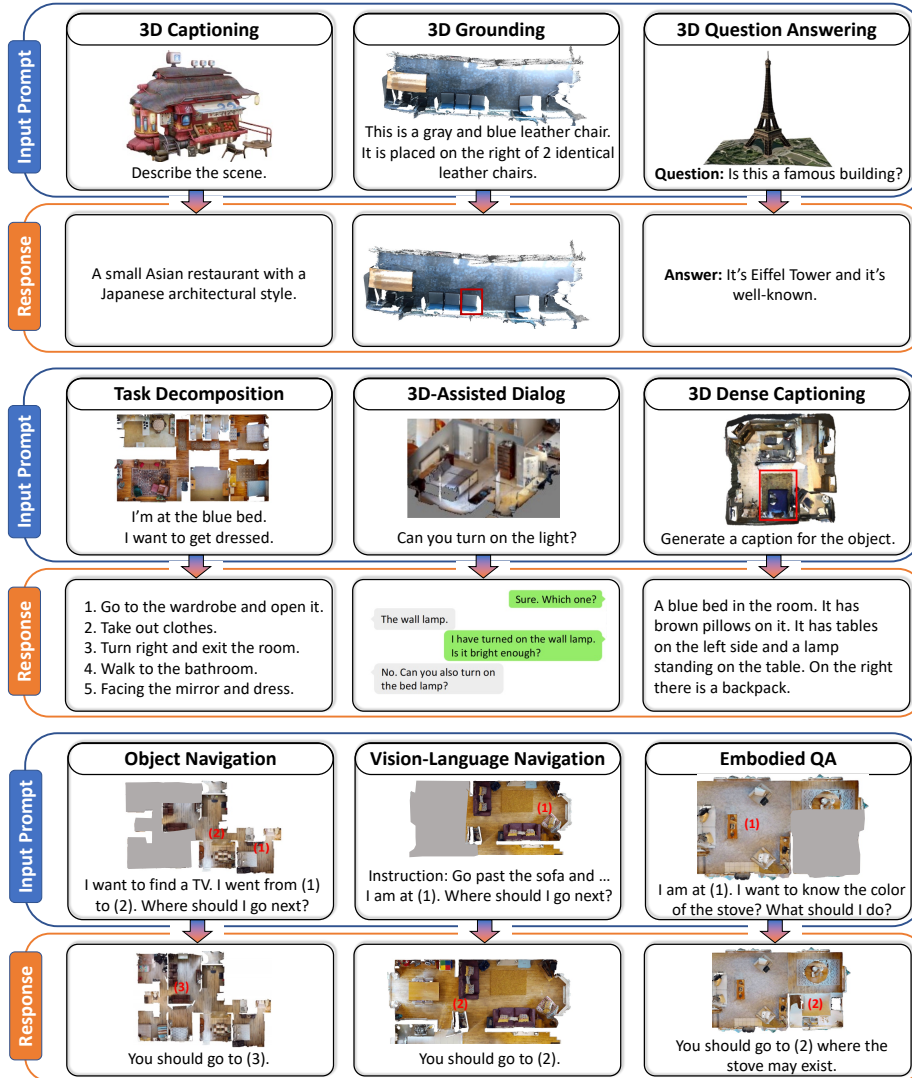


Figure 1: Examples from our generated 3D-language data, which covers multiple 3D-related tasks.

relationships, affordances, physics and interaction so on. Therefore, such LLMs pale in comparison with the robots depicted in sci-fi movies - the assistants that could understand the 3D environments, as well as perform reasoning and planning based on the 3D understandings.

To this end, we propose to inject the 3D world into large language models, and introduce a whole new family of 3D-LLMs that could take 3D representations (*i.e.*, 3D point clouds with their features) as input, and perform a series of 3D-related tasks. By taking the 3D representations of scenes as input, LLMs are blessed with twofold advantages: (1) long-term memories about the entire scene can be stored in the holistic 3D representations, instead of episodic partial-view observations. (2) 3D properties such as affordances and spatial relationships can be reasoned from 3D representations, far beyond the scope of language-based or 2D image-based LLMs.

One major challenge of training the proposed 3D-LLMs lies in data acquisition. Unlike the vast amount of paired 2D-images-and-text data on the Internet, the scarcity of 3D data hinders the development of 3D-based foundation models. 3D data paired with language descriptions are even harder to obtain. To address this, we propose a set of unique data generation pipelines that could generate large-scale 3D data paired with language. Specifically, we make use of ChatGPT [33] and devise three efficient prompting procedures for communication between 3D data and language. In this way, we are able to obtain 300k 3D-language data covering a diverse set of tasks, including but not limited to 3D captioning, dense captioning, 3D question answering, 3D task decomposition, 3D grounding, 3D-assisted dialog, navigation and so on, as shown in Figure 1.

The next challenge resides in how to obtain meaningful 3D features that could align with language features for 3D-LLMs. One way is to train 3D encoders from scratch using a similar contrastive-learning paradigm for the alignment between 2D images and language (*e.g.*, CLIP [37]). However, this paradigm consumes tremendous data, time, and GPU resources. From another perspective, there are numerous recent works that build 3D features from 2D multi-view images (*e.g.*, concept fusion [26], 3D-CLR [20]). Inspired by this, we also utilize a 3D feature extractor that constructs 3D features from the 2D pretrained features of rendered multi-view images. Recently, there are also quite a few visual-language models (*e.g.*, BLIP-2 [30], Flamingo [14]) utilizing the 2D pretrained CLIP features for training their VLMs. Since our extracted 3D features are mapped to the same feature space as 2D pretrained features, we can seamlessly use 2D VLMs as our backbones and input the 3D features for the efficient training of 3D-LLMs.

One crucial aspect of 3D-LLMs, different from vanilla LLMs and 2D VLMs, is that 3D-LLMs are expected to have a underlying 3D spatial sense of information. Thus, we develop a 3D localization mechanism that bridges the gap between language and spatial locations. Specifically, we append 3D position embeddings to the extracted 3D features to better encode spatial information. In addition, we append a series of location tokens to the 3D-LLMs, and localization can be trained via outputting location tokens given the language descriptions of specific objects in the scenes. In this way, 3D-LLMs could better capture 3D spatial information.

To sum up, our paper has the following contributions:

- We introduce a new family of 3D-based Large Language models (3D-LLMs) that can take 3D points with features and language prompts as input, and perform a variety of 3D-related tasks. We focus on tasks beyond the scope of vanilla LLMs or 2D-LLMs, such as tasks about holistic scene understanding, 3D spatial relationships, affordances and 3D planning.
- We devise novel data collection pipelines that could generate large-scale 3D-language data. Based on the pipelines, we collect a dataset that has over 300k 3D-language data that cover a diverse set of 3D-related tasks, including but not limited to 3D captioning, dense captioning, 3D question answering, task decomposition, 3D grounding, 3D-assisted dialog, navigation, and so on.
- We use a 3D feature extractor that extracts meaningful 3D features from rendered multi-view images. We utilize 2D pretrained VLMs as our backbones for efficient training. We introduce a 3D localization mechanism for training the 3D-LLMs to better capture 3D spatial information.
- Experiments on held-out evaluation dataset, ScanQA, outperform state-of-the-art baselines. In particular, 3D LLMs outperform baselines by a large margin on ScanQA (*e.g.*, 9% for BLEU-1). Experiments on held-in datasets for 3D captioning, task composition, and 3D-assisted dialogue show that our model outperforms 2D VLMs. Qualitative studies further demonstrate that our model is able to handle a diverse set of tasks.
- We plan to release our 3D-LLMs, the 3D-language dataset, and language-aligned 3D features of the dataset for future research development.

2 Related Works

Large Language Models. Our work is closely related to large language models [4, 13, 38, 9, 34] (LLMs) like GPT-3 [4] and PaLM [9], which are able to handle different language tasks with a single model and show strong generalization abilities. These models are typically trained on massive textual data with self-supervised training targets like predicting the next tokens [4, 38] or reconstructing the masked tokens [13, 39]. To better align these LLMs’ predictions to human instructions, improve the models’ generalization abilities on unseen tasks, a series of instruction tuning methods [35, 44] and datasets [10, 12] have been proposed. In this work, we aim to inject the 3D world into large language models, understanding rich 3D concepts such as spatial relations, affordances, and physics.

Vision-Language Pre-trained Models. Our work is also related to vision-language pre-trained models that connect images and natural language [31, 32, 17, 37, 27]. Some research [37, 27] learn to train models from scratch with massive image-language pairs and apply them to downstream tasks like visual question answering [18, 51], captioning [7], and referring expression comprehension [50] with finetuning. Other researchers have connected pre-trained vision models and pre-trained LLMs with additional learnable neural modules like perceiver [2] and QFormers [31], leveraging perception abilities in pre-trained vision models, and reasoning and generalization capacities in LLMs. Inspired by these previous works, we plan to build an AI assistant that could understand the 3D world and perform corresponding 3D reasoning and planning. This is not trivial and we need to overcome

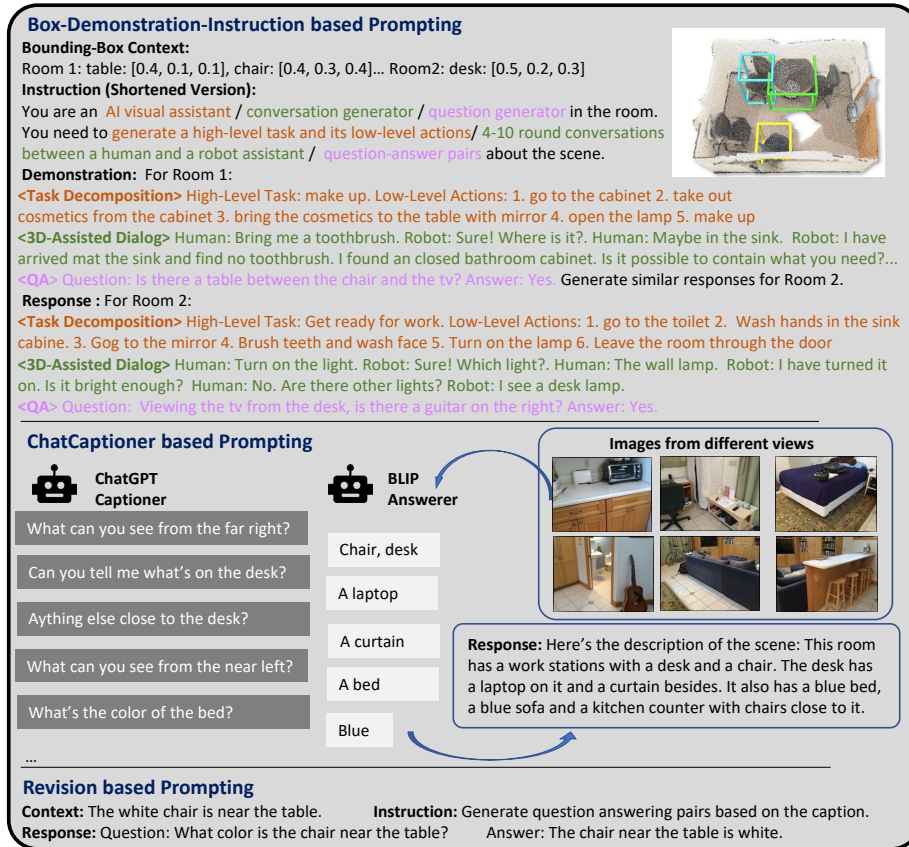


Figure 2: 3D-language data generation pipelines.

obstacles like how to handle the problem of data sparsity, how to align the 3D world with 2D images, and how to capture 3D spatial information.

3D & Language. Another line of research that is similar to ours is 3D and language [5, 49, 8, 20, 1, 15, 24, 49, 3, 21, 19]. ScanQA [49] requires a model to answer questions related to the 3D world; ScanRefer [5] asks a model to localize a region that the text expression refer to; 3D captioning [8] tests models' abilities to generate captions describing the 3D scenes. However, these 3D tasks and their corresponding models are usually task-specific and could only handle cases within the same distribution of the training sets without generalization. Different from them, we aim to build a 3D model that could handle different tasks at the same time and enable new abilities like 3D-assistant dialog and task decomposition.

3 3D-Language Data Generation

The community has witnessed the proliferation of multi-modal data thanks to easy access to a tremendous amount of 2D image and text pairs on the internet. However, when it comes to 3D-related data, obtaining multimodal resource is not easy, due to not only the scarcity of 3D assets, but also the difficulty of providing language data for 3D assets. There are some existing datasets that contain 3D-language data (*e.g.*, ScanQA [49], ScanRefer [5]). However, they are limited with regard to both quantity and diversity, restricted to only one task per dataset. How to generate a 3D-language dataset that can be utilized for all kinds of 3D-related tasks is well worth delving into.

Inspired by the recent success of large language models like GPT [33], we propose to leverage such models for 3D-language data collection. Specifically, as shown in Figure 7, we have three ways to prompt a text-only GPT for generating data. 1) boxes-demonstration-instruction based prompting. We input the axis-aligned bounding boxes (AABB) of both the rooms and the objects in the 3D scenes, providing information about the semantics and spatial locations of the scene. We then provide specific instructions to the GPT model to generate diverse data. We give 0-3 few-shot demonstration examples of the GPT model showing what kind of data it is instructed to generate. 2) ChatCaptioner based prompting. We utilize techniques similar to [52], in which ChatGPT is prompted to ask a series

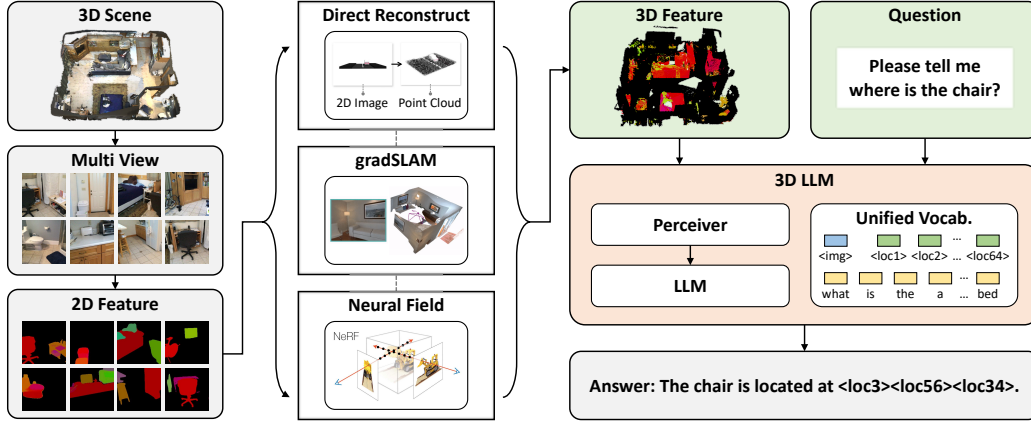


Figure 3: **Overview of our 3D-LLM framework.** The first two columns show our 3D feature extractor. We first render a few multi-view images from the 3D scene, extract 2D dense features, and then construct 3D features from these multi-view images using three kinds of methods. And then, the 3D features and input language prompts are input to the 3D-LLMs to generate responses. We also propose a 3D localization mechanism to better capture 3D spatial information.

of informative questions about an image and BLIP-2 [30] answers the questions. In order to collect 3D-related data, we input images from different views to BLIP-2, and ChatGPT is instructed to ask questions and collect information of different regions to form a global 3D description of the entire scene. 3) Revision based prompting. It can be used for transfer one type of 3D data to another.,

Given the prompting pipelines, GPT is able to generate various types of 3D-language data as summarized in Figure 1. We show detailed prompts to generate all types of data in the Appendix.

We mainly establish our 3D-language dataset upon several 3D assets:

- Objaverse is a universe of 800K 3D objects. However, since the language descriptions were extracted from online sources and not examined by humans, most objects have very noisy descriptions (*e.g.*, with urls) or no descriptions. We utilize ChatCaptioner based prompting to generate high-quality 3D-related descriptions for the scenes.
- Scannet [11] is a richly-annotated dataset of approximately 1k 3D indoor scenes. It provides semantics and bounding boxes of the objects in the scenes.
- Habitat-Matterport (HM3D) [41] is a dataset of 3D environments of embodied AI. HM3DSem [47] further adds semantic annotations and bounding boxes for more than 200 scenes of HM3D.

4 3D-LLM

4.1 Overview

In this section, we introduce how we train our 3D-LLMs. We argue that it’s hard to train 3D-LLMs from scratch, since our collected 3D-language dataset is still not the size of billion-scale image-language dataset used to train 2D VLMs. Furthermore, for 3D scenes, there are no available pretrained encoders like those for 2D images (*e.g.*, CLIP ViT encoders). Thus, retraining 3D-language models from scratch is data-inefficient and resource-heavy. Recently, researchers have proposed to extract 3D features from 2D multi-view images [26, 20]. Using these alignment methods, we could use pretrained image encoders to extract image features, and then map the features to the 3D data. Since the pretrained image features serve as inputs to 2D VLMs, the mapped 3d features of the same feature space can also be seamlessly fed into the pretrained 2D VLMs, which we use as our backbones to train 3D-LLMs. We also propose a 3D localization mechanism to boost the model’s ability to capture 3D spatial information. Figure 3 shows our framework.

4.2 3D Feature Extractor

The first step of training 3D-LLMs is to build meaningful 3D features that could be aligned with language features. For 2D images, there exist feature extractors like CLIP, which learn visual models from language supervision. The models are pretrained using billion-scale internet data of image-language pairs. It’s hard to pre-train such feature learners from scratch, since there are no 3D-language assets comparable to internet-scale image-language pairs in terms of quantity and diversity.

On the contrary, numerous methods have been proposed to extract 3D features from 2D multi-view images [26, 20, 16, 23]. Inspired by these works, we extract features for 3D points by rendering the 3D scenes in several different views, and construct 3D features from rendered image features.

We first extract pixel-aligned dense features for rendered images following [26]. Then, we utilize three methods to construct 3D features from rendered image features. These methods are designed for different types of 3D data.

- **Direct Reconstruction.** We directly reconstruct point cloud from rgbd images rendered from the 3D data using ground-truth camera matrixes. The features are directly mapped to the reconstructed 3D points. This method is suitable for rendered rgbd data with perfect camera poses and intrinsics.
- **Feature Fusion.** Similar to [26], we fuse 2D features into 3D maps using gradslam [28]. Different from dense mapping methods, the features are fused in addition to depths and colors. This method is suitable for 3D data with noisy depth map renderings, or noisy camera poses and intrinsics.
- **Neural Field.** We utilize [20], which constructs 3D compact representation using neural voxel field [43]. Specifically, each voxel in the field has a feature in addition to density and color. Then we align 3D features in the rays and 2D features in the pixels using MSE loss. This method is for 3D data with RGB renderings but no depth data, and noisy camera poses and intrinsics.

In this way, we are able to obtain the $\langle N, \mathcal{D}_v \rangle$ -dim 3D features of each 3D scene, where N is the number of points in the point cloud, and \mathcal{D}_v is the feature dimension.

4.3 Training 3D-LLMs

4.3.1 2D VLMs as backbones

In addition to the feature extractor, training 3D-LLMs from scratch is also non-trivial. In fact, according to [30, 14], the training of 2D VLMs only begins to show "signs of life" after consuming half a billion images. They usually use frozen and pre-trained image encoders such as CLIP to extract features for 2D images. Considering that with 3D feature extractor, the 3D features can be mapped into the same feature space as 2D images, it's reasonable to use these 2D VLMs as our backbones.

The perceiver architecture proposed by [25] leverages an asymmetric attention mechanism to iteratively distill inputs into a tight latent bottleneck, allowing it to handle very large inputs of arbitrary input sizes, thus can tackle different modalities. This architecture is utilized in VLMs like Flamingo [14]. BLIP-2 [30] also utilizes a similar structure called QFormer. The 2D image features, output from frozen image encoders, are flattened and sent to the perceiver to generate a fixed-sized input. Given that our 3D features are in the same feature space as the 2D features by the 3D feature extractor, and that perceiver is able to handle inputs of arbitrary input sizes of the same feature dimension, point cloud features with arbitrary sizes could also be fed into the perceiver. Therefore, we use the 3D feature extractor to extract the 3D features in the same feature space as the features of the frozen image encoders. Then, we use pretrained 2D VLMs as our backbones, input the aligned 3D features to train 3D-LLMs with our collected 3D-language dataset.

4.3.2 3D Localization Mechanism

Apart from building 3D features, which can be aligned with language semantics, it's also essential to capture 3D spatial information. To this end, we propose a 3D localization mechanism that boosts 3D LLMs' abilities to absorb spatial information. It consists of two parts:

Augmenting 3D features with position embeddings Besides the 3D features aggregated from 2D multi-view features, we also add position embeddings to the features. Suppose the feature dim is \mathcal{D}_v . We generate sin/cos position embeddings of the three dimensions, each has an embedding size $\mathcal{D}_v/3$. We concatenate the embeddings of all three dimensions, and concatenate them to the 3D features.

Augmenting LLM vocabularies with location tokens In order to align 3D spatial locations with LLMs, we propose to embed 3D locations in the vocabularies, following [6] and [45]. To be specific, the region to be grounded can be denoted as a sequence of discrete tokens representing the bounding box in the form of AABB. The continuous corner coordinates of the bounding boxes are uniformly discretized to voxel integers as location tokens $\langle x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max} \rangle$. After adding these additional location tokens, we unfreeze the weights for these tokens in the input and output embeddings of language models.

5 Experiments

We first introduce the architecture, and training and evaluation protocols. In Sec 5.1, we analyze the held-out experiments on ScanQA Dataset. Sec 5.2 covers more analysis on held-in evaluation and qualitative examples. Due to page limit, **we put the following content into Appendix**: 1) Held-Out Experiments on 3DMV-VQA and object navigation; 2) Held-In Experiments on grounding and dense captioning; 3) More ablative studies; 4) More qualitative examples.

Architecture We experiment on three backbone 2D VLMs for 3D-LLMs: Flamingo 9B, BLIP-2 Vit-g Opt2.7B, BLIP-2 Vit-g FlanT5-XL. For BLIP-2, during pre-training the 3D-LLMs, we initialize the model from BLIP-2 checkpoints released in LAVIS library [29], and finetune the parameters for the QFormer. 3D features are 1408-dim features, same as EVA_CLIP hidden feature dim used by BLIP-2. We keep most parts of the LLMs (*i.e.*, Opt and FlanT5) frozen, except the weights for the newly-added location tokens in the input and the output embeddings. For Flamingo, we initialize the model from the Flamingo9B checkpoint released in OpenFlamingo repository [2]. We finetune the parameters for perceiver, gated cross attention layers, and the weights for additional location tokens in the input and output embeddings. 3D features are 1024-dim features, same as CLIP hidden feature dim used by Flamingo.

Training & Evaluation Datasets & Protocols We split our datasets into two genres, held-in datasets and held-out datasets. Specifically, our 3D-language data generation pipeline generates the held-in datasets of multiple tasks. we split the datasets into train/val/test sets (8:1:1). We utilize training sets of held-in datasets for pre-training foundation 3D-LLMs, and their validation and test sets can be applied for held-in evaluation. During pre-training, we mix the held-in datasets of all tasks. The models are trained with the standard language modeling loss to output responses. Held-out datasets, on the other hand, are not used in training the foundation 3D-LLMs. We use two held-out 3D question answering datasets for held-out evaluation: ScanQA and 3DMV-VQA. We put the analysis of experiments of 3DMV-VQA[20] in the supplementary material.

5.1 Held-Out Evaluation

We finetune our pretrained 3D-LLMs on the ScanQA dataset and compare with baseline models.

Baselines & Evaluation Metrics We include representative baseline models on the benchmark. Particularly, **ScanQA** is the state-of-the-art method on the benchmark that uses VoteNet to obtain object proposals, and then fuse them with language embeddings. **ScanRefer+MCAN** is a baseline that identifies the referred object and the MCAN model is applied to the image surrounding the localized object. **VoteNet+MCAN** detects objects in a 3D space, extracts their features, and uses them in a standard VQA model. Notably, these baseline models all extract explicit object representations from a pretrained localization module. In addition to these baselines, we also design several LLM-based baselines. **LLaVA** is a visual instruction tuning that connects a vision encoder and LLM for general-purpose visual and language understanding. We use its pretrained model and do zero-shot evaluation on our dataset. We use a single random image as input. We use LLaVA 13B model. **Single Image + Pretrained VLMs** use our 2D VLM backbones (*i.e.*, flamingo and BLIP-2), replace the 3D inputs of 3D-LLMs with single image features to train the models, and then finetune on ScanQA dataset. **Multi-View Image + Pretrained VLMs** use our 2D VLM backbones, replace the 3D inputs of 3D-LLMs with concatenated features of multi-view images to train the models, and then finetune on ScanQA dataset. We report BLEU, ROUGE-L, METEOR, CIDEr for robust answer matching. We also use exact match (EM) metric.

Result Analysis We report our results on ScanQA validation set in Table 1, and results on test set in Table 2. We observe a significant increase in the evaluation metrics. For example, for BLEU-1, our model outperforms the state-of-the-art ScanQA model by $\sim 9\%$ for validation set and $\sim 7\%$ for test set. For CIDEr, we report a $\sim 5\%$ gain compared to ScanQA, and much higher than other 3D-based baselines. These results show that by injecting 3D into LLMs, the models can generate answers that are much more similar to the ground-truth answers. Furthermore, 3D-based baselines use object detectors like VoteNet to segment the objects, and then send per-object features into their models, while our inputs are holistic 3D features without explicit object representations. This shows that our model could perform visual reasoning about objects and their relationships even without explicit object representations. We then examine whether 2D VLMs have the same ability. We find that by taking single-view images or multi-view images as inputs, the performances drop much compared to 3D-LLMs. Specifically, multi-view images also contain information about the whole scene. However,

	B-1	B-2	B-3	B-4	METEOR	ROUHE-L	CIDER	EM
VoteNet+MCAN*	28.0	16.7	10.8	6.2	11.4	29.8	54.7	17.3
ScanRefer+MCAN*	26.9	16.6	11.6	7.9	11.5	30	55.4	18.6
ScanQA*	30.2	20.4	15.1	10.1	13.1	33.3	64.9	21.0
LLaVA(zero-shot)	7.1	2.6	0.9	0.3	10.5	12.3	5.7	0.0
flamingo-SingleImage	23.8	14.5	9.2	8.5	10.7	29.6	52	16.9
flamingo-MultiView	25.6	15.2	9.2	8.4	11.3	31.1	55	18.8
BLIP2-flant5-SingleImage	28.6	15.1	9.0	5.1	10.6	25.8	42.6	13.3
BLIP2-flant5-MultiView	29.7	16.2	9.8	5.9	11.3	26.6	45.7	13.6
3D-LLM (flamingo)	30.3	17.8	12.0	7.2	12.2	32.3	59.2	20.4
3D-LLM (BLIP2-opt)	35.9	22.5	16.0	9.4	13.8	34.0	63.8	19.3
3D-LLM (BLIP2-flant5)	39.3	25.2	18.4	12.0	14.5	35.7	69.4	20.5

Table 1: Experimental results on ScanQA validation set. * Means the models use explicit object representations. B-1, B-2, B-3, B-4 denote BLEU-1, BLEU-2, BLEU-3, BLEU-4 respectively. Our model outperforms all baseline models for all evaluation metrics except for the EM metric.

	BLEU-1	BLEU-4	METEOR	ROUHE-L	CIDER	EM
SingleImage+MCAN	16.5	0.0	8.4	21.5	38.6	15.8
VoteNet+MCAN*	29.5	6.0	12.0	30.9	58.2	19.7
ScanRefer+MCAN*	27.9	7.5	11.9	30.7	57.4	20.6
ScanQA*	31.6	12.0	13.5	34.3	67.3	23.5
3D-LLM (flamingo)	32.6	8.4	13.5	34.8	65.6	23.2
3D-LLM (BLIP2-opt)	37.3	10.7	14.3	34.5	67.1	19.1
3D-LLM (BLIP2-flant5)	38.3	11.6	14.9	35.3	69.6	19.1

Table 2: Experimental results on ScanQA test set. * Means the models use explicit object representations. Our model outperforms all baseline models for most of the evaluation metrics.

they have much lower performances compared to 3D-LLMs, probably because features of multi-view images are disorganized, thus losing 3D-related information.

5.2 More Extensive Evaluation

Held-In Evaluation We carry out experiments on held-in datasets of three tasks: 3D captioning, 3D-assited dialog and task decomposition. The baselines include 2D VLMs as for the held-out evaluation. We add one language-only baseline: Flant5, which examines LLMs’ ability to complete these tasks without any visual input. To evaluate the quality of responses, we include BLEU, ROUGE-L, METEOR, CIDEr as our metrics. We report the held-in evaluation performances in Table 3. From the table, we could see that 3D-LLMs could generate high-quality responses, outperforming both 2D VLMs and language-only LLMs.

Tasks	Models	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGH-L
3D Captioning	flamingo-SingleImage	29.0	17.9	12.5	12.1	12.4	28.2
	flamingo-MultiView	29.5	18.6	13.7	12.4	14.0	29.0
	BLIP2-flant5-SingleImage	30.3	18.3	14.5	12.0	13.1	30.9
	BLIP2-flant5-MultiView	34.4	23.9	18.0	14.1	17.5	35.7
	3D-LLM (flamingo)	36.1	24.5	18.7	15.6	17.6	35.8
	3D-LLM (BLIP2-opt)	35.7	26.7	20.3	15.9	18.7	40.1
	3D-LLM (BLIP2-t5)	39.8	31.0	24.7	20.1	17.7	42.6
3D-assisted Dialog	flant5	27.4	16.5	11.1	8.7	9.5	27.5
	flamingo-SingleImage	29.4	18.7	11.3	9.4	10.0	26.8
	flamingo-MultiView	30.6	21.3	11.9	9.1	10.4	27.9
	BLIP2-flant5-SingleImage	28.4	17.3	10.6	9.1	10.2	27.4
	BLIP2-flant5-MultiView	32.4	20.9	12.1	9.5	11.0	29.5
	3D-LLM (flamingo)	35.0	22.8	15.4	10.6	16.0	34.2
	3D-LLM (BLIP2-opt)	39.6	27.5	20.5	16.2	18.4	38.6
3D-LLM (BLIP2-flant5)	39.0	27.8	21.2	16.6	18.9	39.3	
Task Decomposition	flant5	25.5	21.1	16.7	6.0	13.9	28.4
	flamingo-SingleImage	31.4	23.0	18.8	7.1	15.6	30.6
	flamingo-MultiView	33.1	24.7	21.4	7.3	16.1	33.2
	BLIP2-flant5-SingleImage	32.2	25.3	18.2	6.9	15.0	31.0
	BLIP2-flant5-MultiView	33.1	27.0	20.6	6.9	15.5	34.0
	3D-LLM (flamingo)	32.9	25.6	20.2	6.4	16.0	33.5
3D-LLM (BLIP2-opt)	34.1	27.7	20.8	7.6	16.5	35.4	
3D-LLM (BLIP2-flant5)	33.9	28.1	20.7	7.4	15.9	37.8	

Table 3: Experimental Results on Held-In Datasets. 3D-LLMs outperform 2D VLMs.

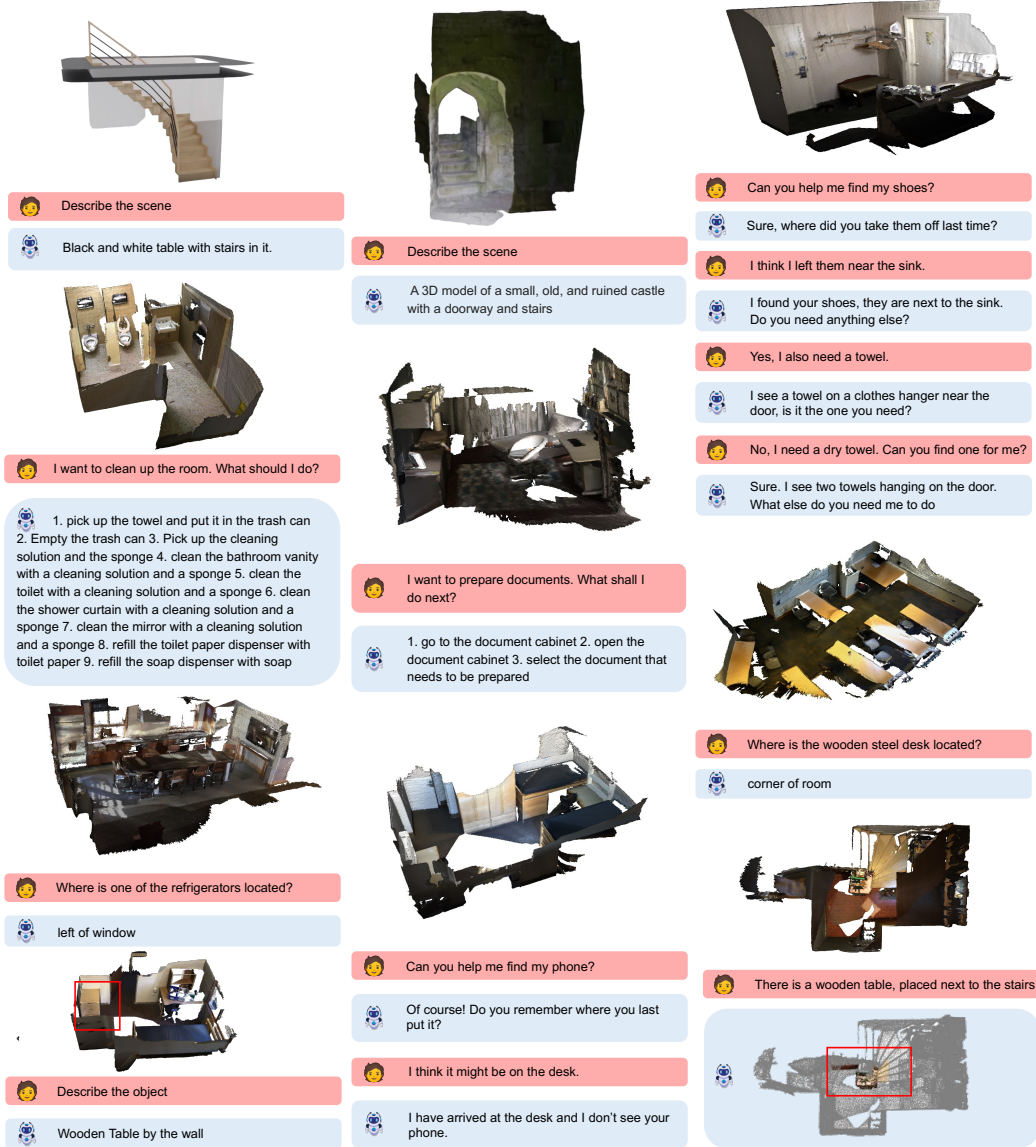


Figure 4: Qualitative examples of 3D-LLM’s prediction.

Qualitative Examples In Figure 4, we show qualitative examples of 3D-LLM’s predictions. We can see that our 3D-LLM is able to perform a variety of tasks.

6 Conclusion

In this paper, we propose a new family of 3D-LLMs that can take 3D representations as inputs and generate responses. We introduce a series of 3D-language data generation pipelines to generate a dataset of 300K 3D-language pairs to train our 3D-LLMs, including dense captioning, 3D question answering, task decomposition, 3D grounding, 3D-assisted dialog, navigation, and so on. Our 3D-LLMs leverage 2D pretrained VLMs as backbones and a novel 3D localization mechanism. Experiments show that our 3D-LLMs outperform state-of-the-art baseline models on ScanQA datasets, and could perform a diverse set of 3D-related tasks. A limitation is that the 3D feature extractor relies on 2D multi-view images, and thus all 3D scenes need to be rendered so that they can be trained in 3D-LLMs, which introduces an additional rendering process.

References

- [1] P. Achlioptas, A. Abdelreheem, F. Xia, M. Elhoseiny, and L. J. Guibas. ReferIt3D: Neural listeners for fine-grained 3D object identification in real-world scenes. In *ECCV*, 2020.
- [2] A. Awadalla, I. Gao, J. Gardner, J. Hessel, Y. Hanafy, W. Zhu, K. Marathe, Y. Bitton, S. Gadre, J. Jitsev, S. Kornblith, P. W. Koh, G. Ilharco, M. Wortsman, and L. Schmidt. Openflamingo, Mar. 2023.
- [3] D. Azuma, T. Miyanishi, S. Kurita, and M. Kawanabe. ScanQA: 3D question answering for spatial scene understanding. *2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19107–19117, 2022.
- [4] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, pages 1877–1901, 2020.
- [5] D. Z. Chen, A. X. Chang, and M. Nießner. ScanRefer: 3D object localization in RGB-D scans using natural language. *16th European Conference on Computer Vision (ECCV)*, 2020.
- [6] T. Chen, S. Saxena, L. Li, D. J. Fleet, and G. E. Hinton. Pix2seq: A language modeling framework for object detection. *ArXiv*, abs/2109.10852, 2021.
- [7] X. Chen, H. Fang, T.-Y. Lin, R. Vedantam, S. Gupta, P. Dollár, and C. L. Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv preprint arXiv:1504.00325*, 2015.
- [8] Z. Chen, A. Gholami, M. Nießner, and A. X. Chang. Scan2cap: Context-aware dense captioning in rgb-d scans. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3193–3203, 2021.
- [9] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.
- [10] H. W. Chung, L. Hou, S. Longpre, B. Zoph, Y. Tay, W. Fedus, E. Li, X. Wang, M. Dehghani, S. Brahma, et al. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*, 2022.
- [11] A. Dai, A. X. Chang, M. Savva, M. Halber, T. A. Funkhouser, and M. Nießner. ScanNet: Richly-annotated 3D reconstructions of indoor scenes. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2432–2443, 2017.
- [12] Databricks. Free dolly: Introducing the world’s first truly open instruction-tuned llm, 2023.
- [13] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [14] J.-B. A. et al. Flamingo: a visual language model for few-shot learning. 2022.
- [15] M. Feng, Z. Li, Q. Li, L. Zhang, X. Zhang, G. Zhu, H. Zhang, Y. Wang, and A. S. Mian. Free-form description guided 3d visual graph network for object grounding in point cloud. *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 3702–3711, 2021.
- [16] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song. CLIP on wheels: Zero-shot object navigation as object localization and exploration. *ArXiv*, abs/2203.10421, 2022.
- [17] T. Gong, C. Lyu, S. Zhang, Y. Wang, M. Zheng, Q. Zhao, K. Liu, W. Zhang, P. Luo, and K. Chen. MultiModal-GPT: A vision and language model for dialogue with humans. *arXiv preprint arXiv:2305.04790*, 2023.
- [18] Y. Goyal, T. Khot, D. Summers-Stay, D. Batra, and D. Parikh. Making the V in VQA matter: Elevating the role of image understanding in Visual Question Answering. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.
- [19] Y. Hong, Y. Du, C. Lin, J. Tenenbaum, and C. Gan. 3d concept grounding on neural fields. *Advances in Neural Information Processing Systems*, 35:7769–7782, 2022.
- [20] Y. Hong, C. Lin, Y. Du, Z. Chen, J. B. Tenenbaum, and C. Gan. 3D concept learning and reasoning from multi-view images, 2023.

- [21] Y. Hong, C. Lin, Y. Du, Z. Chen, J. B. Tenenbaum, and C. Gan. 3d concept learning and reasoning from multi-view images. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023.
- [22] R. Hu, H. Xu, M. Rohrbach, J. Feng, K. Saenko, and T. Darrell. Natural language object retrieval, 2016.
- [23] C. Huang, O. Mees, A. Zeng, and W. Burgard. Visual language maps for robot navigation, 2023.
- [24] P.-H. Huang, H.-H. Lee, H.-T. Chen, and T.-L. Liu. Text-guided graph neural networks for referring 3D instance segmentation. In *AAAI*, 2021.
- [25] A. Jaegle, F. Gimeno, A. Brock, A. Zisserman, O. Vinyals, and J. Carreira. Perceiver: General perception with iterative attention. In *International Conference on Machine Learning*, 2021.
- [26] K. M. Jatavallabhula, A. Kuwajerwala, Q. Gu, M. Omama, T. Chen, S. Li, G. Iyer, S. Saryazdi, N. Keetha, A. Tewari, J. B. Tenenbaum, C. M. de Melo, M. Krishna, L. Paull, F. Shkurti, and A. Torralba. Conceptfusion: Open-set multimodal 3D mapping, 2023.
- [27] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *International Conference on Machine Learning*, pages 4904–4916. PMLR, 2021.
- [28] J. Krishna Murthy, S. Saryazdi, G. Iyer, and L. Paull. gradslam: Dense slam meets automatic differentiation. *arXiv*, 2020.
- [29] D. Li, J. Li, H. Le, G. Wang, S. Savarese, and S. C. H. Hoi. LAVIS: A library for language-vision intelligence, 2022.
- [30] J. Li, D. Li, S. Savarese, and S. Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models, 2023.
- [31] J. Li, D. Li, S. Savarese, and S. Hoi. BLIP-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.
- [32] H. Liu, C. Li, Q. Wu, and Y. J. Lee. Visual instruction tuning. *arXiv preprint arXiv:2304.08485*, 2023.
- [33] OpenAI. GPT-4 technical report, 2023.
- [34] OpenAI. GPT-4 technical report. *ArXiv*, abs/2303.08774, 2023.
- [35] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744, 2022.
- [36] C. Qi, O. Litany, K. He, and L. J. Guibas. Deep hough voting for 3d object detection in point clouds. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9276–9285, 2019.
- [37] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.
- [38] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.
- [39] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.
- [40] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang, et al. Habitat-matterport 3d dataset (hm3d): 1000 large-scale 3d environments for embodied ai. *arXiv preprint arXiv:2109.08238*, 2021.
- [41] S. K. Ramakrishnan, A. Gokaslan, E. Wijmans, O. Maksymets, A. Clegg, J. M. Turner, E. Undersander, W. Galuba, A. Westbury, A. X. Chang, M. Savva, Y. Zhao, and D. Batra. Habitat-matterport 3D dataset (HM3D): 1000 large-scale 3D environments for embodied AI. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021.
- [42] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, et al. Habitat: A platform for embodied ai research. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9339–9347, 2019.

- [43] C. Sun, M. Sun, and H.-T. Chen. Direct voxel grid optimization: Super-fast convergence for radiance fields reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5459–5469, June 2022.
- [44] Z. Sun, Y. Shen, Q. Zhou, H. Zhang, Z. Chen, D. Cox, Y. Yang, and C. Gan. Principle-driven self-alignment of language models from scratch with minimal human supervision. *arXiv e-prints*, pages arXiv–2305, 2023.
- [45] P. Wang, A. Yang, R. Men, J. Lin, S. Bai, Z. Li, J. Ma, C. Zhou, J. Zhou, and H. Yang. Unifying architectures, tasks, and modalities through a simple sequence-to-sequence learning framework. In *International Conference on Machine Learning*, 2022.
- [46] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.
- [47] K. Yadav, R. Ramrakhya, S. K. Ramakrishnan, T. Gervet, J. Turner, A. Gokaslan, N. Maestre, A. X. Chang, D. Batra, M. Savva, et al. Habitat-matterport 3D semantics dataset. *arXiv preprint arXiv:2210.05633*, 2022.
- [48] Z. Yang, B. Gong, L. Wang, W. Huang, D. Yu, and J. Luo. A fast and accurate one-stage approach to visual grounding, 2019.
- [49] S. Ye, D. Chen, S. Han, and J. Liao. 3D question answering. *IEEE transactions on visualization and computer graphics*, PP, 2021.
- [50] L. Yu, P. Poirson, S. Yang, A. C. Berg, and T. L. Berg. Modeling context in referring expressions. In *Computer Vision—ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11–14, 2016, Proceedings, Part II 14*, pages 69–85. Springer, 2016.
- [51] P. Zhang, Y. Goyal, D. Summers-Stay, D. Batra, and D. Parikh. Yin and Yang: Balancing and answering binary visual questions. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [52] D. Zhu, J. Chen, K. Haydarov, X. Shen, W. Zhang, and M. Elhoseiny. Chatgpt asks, blip-2 answers: Automatic questioning towards enriched visual descriptions, 2023.

A 3D-Language Data

A.1 Prompts

In figure 5 and 6, we show two exemplar prompts for generating task decomposition data and 3D-assisted dialog data. Specifically, they are generated using the boxes-demonstration-instruction method described in the paper. For each sample in the few shot samples, the "content" has the bounding boxes of the scenes, and the "response" refers to human-written responses for demonstration. For query, it consists of the bounding boxes of scenes that we query the ChatGPT to give us responses.

```
messages=[{"role": "system", "content": "You are an AI visual assistant that can analyze a 3D scene. All object instances in this 3D scene are given, along with their center point position. The center points are represented by a 3D coordinate (x, y, z) with units of meters. Using the provided object instance information, design a high-level task that can be performed in this 3D scene. Besides, decomposing this high-level task into a sequence of action steps that can be performed using the instances in this 3D scene. \n\n Remember, the high-level task and action steps must be able to be performed in the 3D scene using the given object instances. "}]  
for sample in fewshot_samples:  
    messages.append({"role": "user", "content": sample['content']})  
    messages.append({"role": "assistant", "content": sample['response']})  
messages.append({"role": "user", "content": '\n'.join(query)})
```

Figure 5: Prompts on generating task decomposition data

```
messages=[{"role": "system", "content": "You are a conversation generator in a room. All object instances in this room are given, along with their center point position. The center points are represent by a 3D coordinate (x, y, z) with units of meters. You need to generate 4~10 round conversation between a human and a robot assistant. \n\nThe human know all information in this room, including all objects described above and all small things that are not visible now. The human will ask the robot to do a high-level task. The robot will tell its observation and its state (e.g., location) to the human and will ask for help when it is ambiguous about the task. Remember, the high-level task should be done in this room. "}]  
  
for sample in fewshot_samples:  
    messages.append({"role": "user", "content": sample['content']})  
    messages.append({"role": "assistant", "content": sample['response']})  
messages.append({"role": "user", "content": '\n'.join(query)})
```

Figure 6: Prompts on generating 3D-assisted dialog data

A.2 Data Distribution

In figure 7, we show the distribution of our data.

B Experiments

B.1 Implementation Details

Using Pretrained BLIP-2 as backbones, we train 3D-LLMs for 100K steps, and validate every 1K step. We run the models on 8 nodes, where each node has 8 V100s. The batch size is 16 for each node. The AdamW optimizer is used, with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and a weight decay of 0.05. Additionally, we apply a linear warmup of the learning rate during the initial 1K steps, increasing from 10^{-8} to 10^{-5} , followed by a cosine decay with a minimum learning rate of 0.

3D-LLMs based on pretrained flamingo are trained using the AdamW optimizer with global norm clipping of 1, no weight decay for the perceiver resampler and weight decay of 0.1 for the other

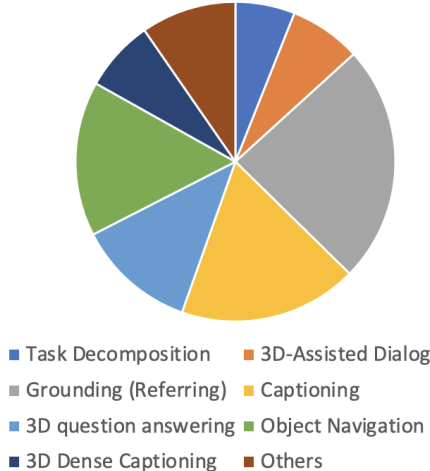


Figure 7: Data distribution on different types for our 3D-language data

trainable parameters. The learning rate is increased linearly from 0 to 10^{-4} up over the first 5000 steps then held constant for the duration of training. The model is trained on 8 A100s. The batch size is 16. We use Distributed Data Parallel (DDP) to train the model.

B.2 Held-Out Evaluation

B.2.1 3DMV-VQA

We finetune our pretrained 3D-LLMs on the 3DMV-VQA dataset and compare with baseline models.

Baselines & Evaluation Metrics We include representative baseline models on the benchmark. **NS-VQA** is the neuro-symbolic method that first extracts object proposals and then perform neuro-symbolic reasoning. **3D-Feature + LSTM** is a baseline that extracts 3D features first and then concatenates with LSTM to output final answers. **3D-CLR** is the state-of-the-art method that extracts 3D feature fields first, and then perform neuro-symbolic reasoning. In addition to these baselines, we also add 2D VLMs baselines like we did for ScanQA.

Methods	Concept	Counting	Relation	Comparison	Overall
NS-VQA*	59.8	21.5	33.4	61.6	38.0
3D-Feature+LSTM	61.2	22.4	49.9	61.3	48.2
3D-CLR*	66.1	41.3	57.6	72.3	57.7
flamingo-SingleImage	58.7	18.5	38.4	60.1	40.3
flamingo-MultiView	60.0	18.3	40.2	61.4	41.6
BLIP-SingleImage	58.0	20.4	42.3	62.3	43.1
BLIP-MultiView	61.9	21.1	48.0	62.3	47.1
3D-LLM (flamingo)	68.9	32.4	61.6	68.3	58.6
3D-LLM (BLIP5-opt)	63.4	30.7	57.6	65.2	54.9
3D-LLM (BLIP2-flanT5)	68.1	31.4	55.1	69.7	54.6

Table 4: Experimental results on 3DMV-VQA dataset. * denotes using explicit object representations and neuro-symbolic reasoning.

Result Analysis Table 4 shows the performances on 3DMV-VQA. We can see that 3D-LLMs outperform state-of-the-art baseline model in the question types of concept and relation, and also in the overall performance. Our model also outperforms 3D-Feature+LSTM, demonstrating the power of LLMs over vanilla language models with similar 3D features as inputs. Overall, 3D-based methods outshine 2D-based versions of the methods. Our 3D-LLMs outperform their corresponding 2D VLMs with image input, further demonstrating the importance of 3D representations for 3D-LLMs.

B.2.2 3D Grounding (Referring) on ScanRefer

In order to examine 3D-LLMs’ 3D localization abilities, we carry out a held-out experiment on ScanRefer benchmark. Specifically, ScanRefer benchmark requires the models to output object locations given a referring sentence of the objects. We finetune 3D-LLMs on ScanRefer training set and report the results on ScanRefer validation sets.

Baselines We include the baseline models in the original ScanRefer paper. Specifically, **OCRand(OracleCatRand)** use an oracle with ground truth bounding boxes of objects, and predict the box by simply selecting a random box that matches the object category. **Vote+Rand(VoteNetRand)** uses the predicted object proposals of the VoteNet [36] backbone and selects a box randomly with the correct semantic class label. **SCRC & One-stage** are 2D image baselines for referring expression comprehension by extending SCRC [22] and One-stage [48] to 3D using back-projection. Since 2D referring expression methods operate on a single image frame, we construct a 2D training set by using the recorded camera pose associated with each annotation to retrieve the frame from the scan video with the closest camera pose. At inference time, we sample frames from the scans (using every 20th frame) and predict the target 2D bounding boxes in each frame. We then select the 2D bounding box with the highest confidence score from the bounding box candidates and project it to 3D using the depth map for that frame. **ScanRefer** uses a pretrained VoteNet backbone with a trained GRU for selecting a matching bounding box.

Evaluation Metrics To evaluate the performance of our method, we measure the thresholded accuracy where the positive predictions have higher intersection over union (IoU) with the ground truths than the thresholds. Similar to work with 2D images, we use ACC@kIoU as our metric, where the threshold value k for IoU is set to 0.25. In addition to that, we also report the Average IoUs of our 3D-LLMs. Since our model focuses on 3D localization of objects, we also report the distances from the centers of predicted bounding boxes to the ground-truth bounding boxes.

Result Analysis In Table 5, we show the results on ScanRefer. As we can see, our 3D-LLMs can have decent performances on grounding and referring, and outperform most of the baselines, showing that 3D-LLMs have the ability of 3D localization. Notably, the baseline models use ground-truth bounding boxes, or a pre-trained object detector to propose bounding boxes and classes for object proposals. Then, they use scoring modules to vote for the most likely candidate. Our method does not use any explicit object proposal module or ground truth bounding boxes, but outputs the locations of the bounding boxes directly using LLM losses for predicting tokens, while still outperforming most of the baselines. We could also see from the Avg. Dist metric the bounding boxes we predict is very close to the ground-truth bounding boxes.

	OCRand	Vote+Rand	SCRC	One-stage	ScanRefer	3D-LLM (flamingo)	3D-LLM (BLIP2-opt)	3D-LLM (BLIP2-flant5)
ACC@0.25	29.9	10.0	18.7	20.4	41.2	21.2	29.6	30.3
Avg. IoU	/	/	/	/	/	19.9	23.1	24.9
Avg. Dist.	/	/	/	/	/	1.1	1.07	1.03

Table 5: Experimental Results on ScanRefer

B.2.3 Object Navigation

We show the ability of our 3D-LLM to progressively understand the environment and navigate to a target object. We formulate the navigation process as a conversation. At each time step, we online build a 3D feature from the partially observed scene. We feed this feature, current agent location, and history location to the 3D-LLM for predicting a 3D waypoint the agent should go for. We then use an off-the-shelf local policy [46] to determine a low-level action (*e.g.*, go forward, turn left or right) for navigating to the waypoint. The 3D-LLM predicts “stop” if it believes the agent has reached the target object.

In Figure 8, we visualize a conversation process and its corresponding navigation trajectory. At the beginning when the target object is not observed, the 3D-LLM predicts a waypoint that leads the agent to explore the area most likely containing the target object. When the agent observes the target object (*i.e.*, red box in the partially observed scene), the 3D-LLM predicts a waypoint leading the agent to it. The example episode is performed on the HM3D dataset [40] using Habitat simulator [42].

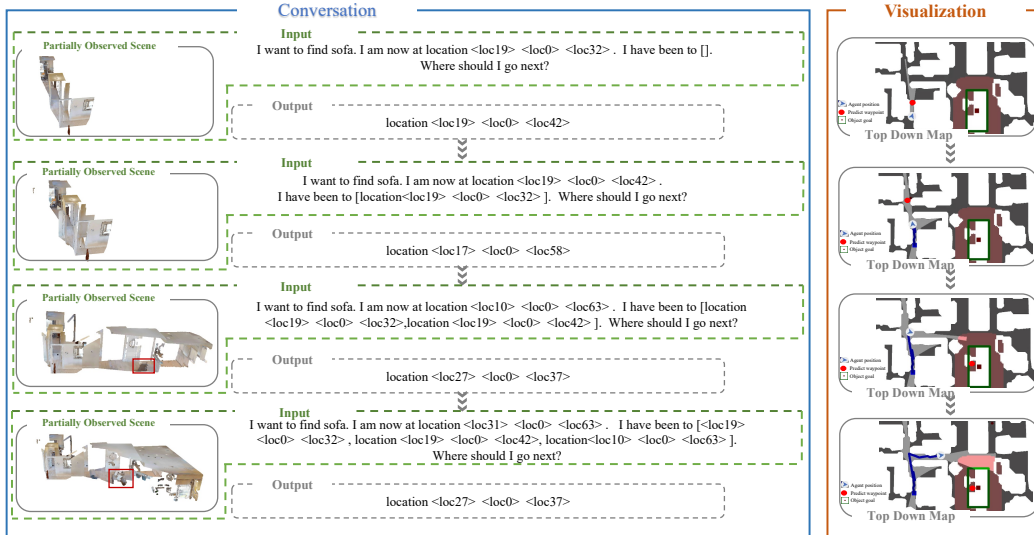


Figure 8: Visualization of an object navigation episode.

B.3 Held-In Evaluation

B.3.1 3D Dense Captioning

In Table 6, we show the results of 3D dense captioning. Specifically, given a 3D bounding box, models are expected to output the caption describing what’s in that region. We can see that our 3D-LLMs outperform image-based baselines.

	BLEU-1	BLEU-2	BLEU-3	BLEU4	METEOR	ROUGH-L
flamingo-SingleImage	21.5	10.5	6.9	4.1	11.1	23.4
flamingo-MultiView	24.4	12.3	7.1	4.6	11.9	25.8
BLIP-SingleImage	23.0	11.7	7.7	4.6	11.3	23.8
BLIP-MultiView	25.3	14.1	9.0	5.6	12.5	24.9
3D-LLM (flamingo)	29.6	16.8	10.6	5.9	11.4	29.9
3D-LLM (BLIP2-opt)	32.5	18.7	11.9	6.5	11.7	31.5
3D-LLM (BLIP2-flant5)	34.3	20.5	13.2	8.1	13.1	33.2

Table 6: Experimental Results on Held-In 3D Dense Captioning Dataset.

B.4 More Ablative Studies

B.4.1 Ablative Studies on Flamingo Perceiver

We first examine how the perceiver resampler of Flamingo benefits the training. We carry out an ablative experiment where we take out the perceiver of the flamingo model. Table 7 shows the results. From the table, we can see that the perceiver module is indeed beneficial for the training of 3D-LLM.

	BLEU-1	BLEU-2	BLEU-3	BLEU-4	METEOR	ROUGH_L	CIDER	EM
wo/ perceiver	29.2	17.2	11.2	7.4	11.4	30.4	58.9	20.6
w/ perceiver	30.3	17.8	12	7.2	12.2	32.3	59.2	20.4

Table 7: Ablative Study on the Perceiver of Flamingo Model.

B.5 More Qualitative Examples

We show more qualitative examples in Figure 9, 10, 11.

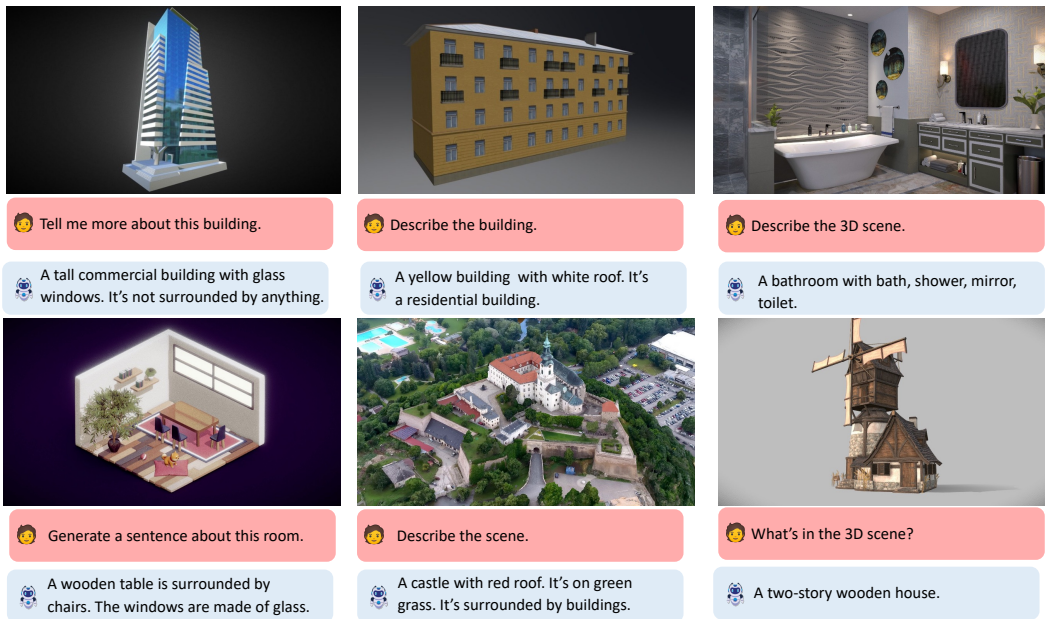


Figure 9: Qualitative Examples on 3D Captioning

		
<p>Can you help me find my book?</p> <p>Sure. Where do you think you left it?</p> <p>I think I left it on the desk.</p> <p>I have arrived at the desk. I see a few books here. Which one do you need?</p>	<p>Can you help me prepare dinner?</p> <p>Sure! What do you want to make?</p> <p>I want to make spaghetti carbonara. Can you help me find the ingredients?</p> <p>I have the blue box of spaghetti and eggs from the close doors. Anything else?</p>	<p>Can you help me find my phone?</p> <p>Sure! Where do you usually keep it?</p> <p>It's usually on my desk.</p> <p>I'm at the desk and don't see a phone. Is there any other place you might keep it?</p> <p>Sometimes I leave it on the bed.</p> <p>I found it under the pillow. Anything else you need?</p>
		
<p>Describe the object.</p> <p>It's a brown cabinet.</p>	<p>What's this?</p> <p>A picture on the wall.</p>	<p>What's in the region?</p> <p>A white square box.</p>
		
<p>Where's the chair located?</p> <p>Front of desk.</p>	<p>What's the style of the building?</p> <p>It's Chinese style.</p>	<p>What's the color of the bathroom stall door?</p> <p>White.</p>
		
<p>What's the color of the building?</p> <p>White.</p>	<p>What's above the sink?</p> <p>A mirror.</p>	<p>How many shelves does the bookshelf have?</p> <p>2.</p>
		
<p>I want to clean up the room. What should I do?</p> <p>1. go to the trash can 2. empty the trash 3. go to the paper towel dispenser 4. refill the paper towels 5. go to the chair 6. clean the chair 7 go to the table 8. clean the table 9. go to the door 10. clean the door 11. go to the window 12. clean the window 13. sweep and mop the floor.</p>	<p>I want to prepare for a meeting. What should I do?</p> <p>1. enter the room through the door 2. go to the cabinets and take out the documents 3. go to the table and arrange the documents 4. exit the room through the door</p>	<p>I want to prepare a cup of coffee. What should I do?</p> <p>1. go to the kitchen counter 2. take the coffee maker 3. fill water from the sink 4. add coffee to the coffee maker 5. wait for the coffee</p>
		
<p>This is a gray cabinet. It's to the left of a white cabinet.</p>	<p>The desk is against the wall. there are many objects on it and a chair in front of it.</p>	<p>The desk is against the wall. there are many objects on it and a chair in front of it.</p>
		

Figure 10: Qualitative Examples on 3D-Assisted Dialog, 3D Dense Captioning and Question Answering.

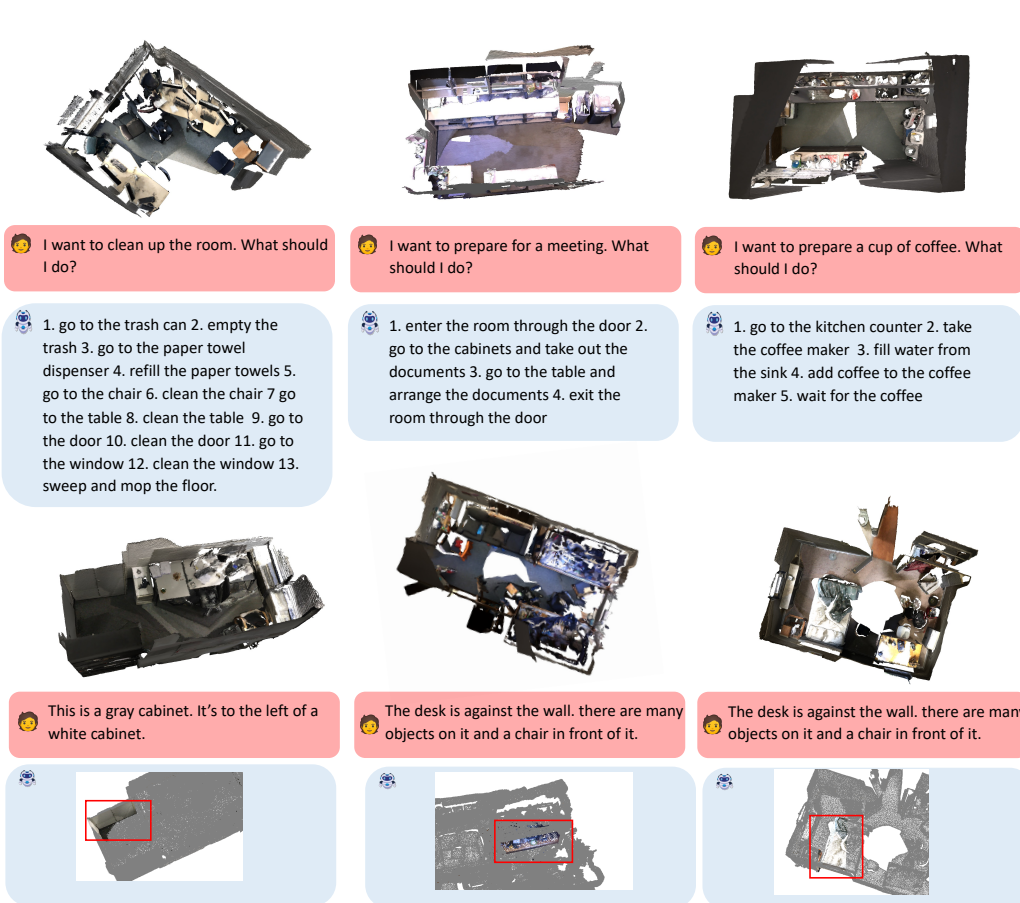


Figure 11: Qualitative Examples on Task Decomposition and Grounding (Referring).