

Transcrib3D: 3D Referring Expression Resolution through Large Language Models

Jiading Fang^{*,1}, Xiangshan Tan^{*,1}, Shengjie Lin^{*,1}, Igor Vasiljevic², Vitor Guizilini²,
Hongyuan Mei¹, Rares Ambrus², Gregory Shakhnarovich^{1,2}, Matthew R. Walter¹

¹Toyota Technological Institute at Chicago ²Toyota Research Institute

Abstract—If robots are to work effectively alongside people, they must be able to interpret natural language references to objects in their 3D environment. Understanding 3D referring expressions is challenging—it requires the ability to both parse the 3D structure of the scene and correctly ground free-form language in the presence of distraction and clutter. We introduce Transcrib3D, an approach that brings together 3D detection methods and the emergent reasoning capabilities of large language models (LLMs). Transcrib3D uses *text* as the unifying medium, which allows us to sidestep the need to learn shared representations connecting multi-modal inputs, which would require massive amounts of annotated 3D data. As a demonstration of its effectiveness, Transcrib3D achieves state-of-the-art results on 3D reference resolution benchmarks, with a great leap in performance from previous multi-modality baselines. To improve upon zero-shot performance and facilitate local deployment on edge computers and robots, we propose self-correction for fine-tuning that trains smaller models, resulting in performance close to that of large models. We show that our method enables a real robot to perform pick-and-place tasks given queries that contain challenging referring expressions. Code will be available at <https://ripl.github.io/Transcrib3D>.

I. INTRODUCTION

Comprehending a natural language expression that mentions an object within a given environment is a routine activity for humans. It often occurs as part of a question (e.g., “Whose jacket is hanging on the black chair?”) or an instruction (e.g., “Pass me the smaller yellow mug.”). This capability is important for embodied agents that work with humans to accomplish tasks. While humans excel in this task with over 90% accuracy on existing benchmarks [1], current computational methods still yield only moderate performance, typically around 60% [2–5]. These methods often rely on learning on a joint multi-modal embedding, where data from different modalities (e.g., point cloud, images, bounding boxes, and text) are combined into a unified representation before being decoded into an answer.

Although this approach is theoretically ideal and shows promise, the process of effectively bridging different modalities in a latent space presents significant challenges. For instance, state-of-the-art 2D vision-language understanding models like CLIP [6] require billions of image-text pairs for training, yet still exhibiting a limited grasp of compositional and relational concepts [7, 8]. The challenge intensifies in 3D domains, where annotated data is much more scarce, with

*Denotes equal contribution.

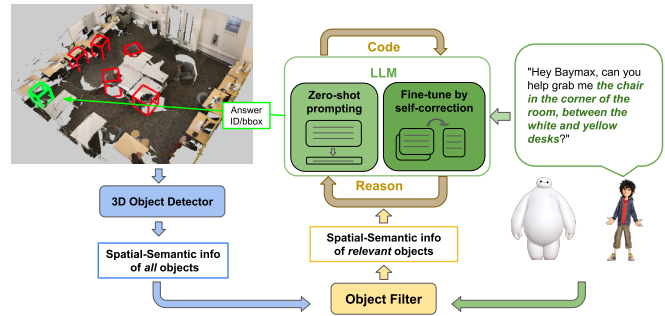


Fig. 1: The overall **Transcrib3D** framework, which takes as input the colored point-cloud and referring expression (in green), and outputs the ID or bounding box of the referent object. To resolve the referring expression “the chair in the corner of the room, between the white and yellow desks”, the framework needs to locate the pillow in the green box, while all other pillows in red boxes are distractors.

more than 500 times difference in scale [9, 10]. As a result, the capacity of existing models to perform 3D reference resolution is limited.

On the contrary, philosophers such as Ludwig Wittgenstein argue that our understanding of reality is confined by the language we use, who famously stated, “The limits of my language mean the limits of my world.” This concept underpins our approach, wherein we propose to employ *text* as the unifying medium to bridge the gap between 3D scene parsing and referential reasoning. This approach is grounded in realizing that the challenge of resolving 3D referring expressions can be divided into two components: *detection* (identifying objects in the scene) and *reasoning* (associating one of the candidates with the referring expression). By harnessing text as a cohesive bridge, we can capitalize on the recent advancements in 3D detection and the enhanced natural language reasoning abilities offered by large language models (LLMs) [11].

Specifically, from the results of an off-the-shelf 3D detector [12], Transcrib3D first converts the detected spatial and semantic 3D scene information—the category, location, spatial extent, and color of objects—into texts, thereby creating an object-centric 3D scene *transcript* (hence the name of our method, **Transcrib3D**). We then filter out non-relevant objects in regard to the query (e.g., information about a trash can is not relevant to the expression “the white pillow on top of the chair and next to a blue pillow”). Subsequently, we

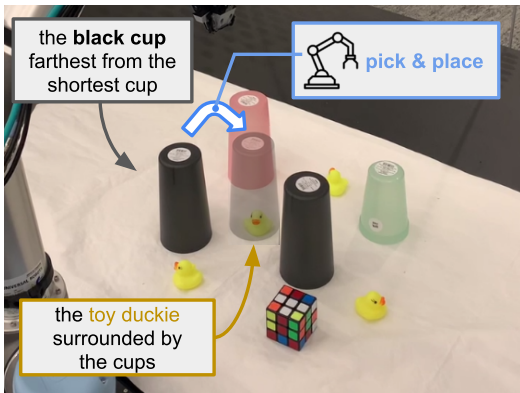


Fig. 2: Transcrib3D enables a robot to resolve complex 3D referring expressions necessary to follow pick-and-place instructions. In this example, the robot is tasked with a natural language instruction that includes challenging referring expressions “cover *the toy duckie surrounded by the cups* with *the black cup farthest from the shortest cup*”.

compose a prompt that incorporates the filtered 3D transcript and the referring expression, and process it through an LLM-based reasoning mechanism. The reasoning module incorporates three key elements to make LLMs more effective and generalizable for the task: (1) Iterative code generation and reasoning; (2) Principle-guided zero-shot prompting; and (3) Fine-tuning from self-reasoned correction.

We evaluate our method on standard 3D referring expression resolution benchmarks, ReferIt3D [1] and ScanRefer [13], achieving state-of-the-art performance on both. We also perform real robot experiments (Fig. 2) that task a robot manipulator with following natural language commands that require sophisticated 3D spatial-semantic reasoning, demonstrating the practicality of the method.

II. RELATED WORK

The problem of resolving 3D referring expressions has garnered significant attention of late, in large part due to the introduction of the ReferIt3D [1] and ScanRefer [13] benchmarks. Contemporary methods [2, 4, 5, 14] perform 3D referring expression resolution by aggregating different input modalities into contextualized embeddings using Transformer architectures [15] in an end-to-end fashion. MVT [2] projects 3D information into 2D to achieve better feature encoding. BUTD-DETR [5] fine-tunes detected 3D bounding boxes within the Transformer. SAT [4] uses 2D semantics during training to learn a mapping from the query to its 3D grounding. HAM [14] presents a hierarchical alignment model that learns multi-granularity visual and linguistic representations. Different from these methods, NS3D [16] proposes a neuro-symbolic framework that utilizes a language-to-code model to generate programs, where each module is represented by neural networks. D3Net [17] and 3DJCG [18] jointly learn 3D captioning and grounding together, where D3Net proposes self-critical training while 3DJCG proposes task-agnostic shared modules and separate task-specific heads. ViL3DRel [19] designs a spatial self-

attention layer to account for relative distances and orientations between 3D objects. 3D-VisTA [3] performs pre-training on a dataset of 278k 3D scene-text pairs, and fine-tunes the model on specific tasks.

In contrast, we propose to connect 3D detection and the LLM reasoning module via a textual representation, which spares us from learning the joint representation of different input modalities from limited 3D annotated data.

A. Grounding Large Language Models

LLMs trained on Internet-scale text data have shown dominant performance across various NLP tasks [11]. However, LLMs have to be grounded to answer questions or execute actions in the physical world. VisProg [20] uses in-context learning for LLMs to generate code for 2D image processing tasks. SayCan [21] and SayPlan [22] instead modulate the LLM outputs with a model of the perceived environment (i.e., “affordances” for SayCan and scene graphs for SayPlan).

When it comes to explicitly reason over 3D inputs, one branch of works train multi-modal models that directly incorporate 3D representations into the token library. 3D-LLM [23] incorporates distilled 3D features [24] with language tokens. 3D-VisTA [3] performs multi-modal fusion of language and PointNet++ features [25] with self-supervised masked encoding training.

Instead of multi-modal approaches, studies have investigated the use of textual 3D information to enhance downstream tasks. For instance, Feng et al. [26] employ textual 3D data in conjunction with LLMs to generate 3D indoor scenes. Similarly, Yu et al. [27] use LLMs for multi-robot navigation, leveraging extracted 3D scene information. We see this body of work as indicative of the promise of textual 3D scene information, and a motivation for our approach. Concurrent with our work, Yang et al. [28] also use LLMs to reason over textual data, however, their methodology differs in its use of a three-step, task-specific reasoning process as opposed to our general, flexible approach.

B. LLM Reasoning

There are many techniques proposed to enhance the reasoning capabilities of LLMs [29–31]. The reasoning module of our method is similar in spirit to the general framework of ReAct [32], where each round of code generation is followed by a round of LLM analysis to proceed with the reasoning or debugging, although our context is 3D-specific. Self-correction is also studied in LLM community. In contrast to Reflexion [33], which uses test-time environmental feedback, we instead collect feedback from training samples, and use the corrected samples to fine-tune models.

III. METHODOLOGY

Figure 1 illustrates our proposed Transcrib3D framework. Given the input colored point-cloud, Transcrib3D first applies a 3D object detector to generate an exhaustive list of objects in the scene transcribed as text (Section III-A). The list is then filtered to identify objects relevant to the provided referring expression (Section III-B). The resulting filtered object information along with the referring expression serve as

input to the LLM-based reasoning module for inference. To reach the final answer, the reasoning module includes a “code interpreter” mode in which the LLM iterates between code generation and reasoning over outputs from code execution (Section III-C). We consider two options for interfacing with the LLM: 1) zero-shot with principles-guided prompting, and 2) fine-tuning from self-reasoned correction (Section III-E).

A. Detect and Transcribe 3D Information

Given a colored point-cloud of the scene, we first perform semantic segmentation of objects using Mask3D [12]. We associate with each detected object its category based on the semantics, center location, spatial extent according to the 3D bounding box, as well as its mean color. The 3D orientation of an object can also be incorporated using PartNet [34]. Transcrib3D compiles the information associated with all detected objects as a list to form an object-centric scene description. The following shows an example of such a scene transcript (exact numeric values and duplicate object categories are replaced by ...).

Spatial Semantic Scene Description for All Objects

```
scene0592: ... Scene center: [...] ... objs list:
monitor, id=0, ctr=[...], size=[...], rgb=[...];
...
box, id=5, ctr=[...], size=[...], rgb=[...];
copier, id=6, ctr=[...], size=[...], rgb=[...];
floor, id=7, ctr=[...], size=[...], rgb=[...];
wall, id=8, ctr=[...], size=[...], rgb=[...];
...
armchair, id=15, ctr=[...], size=[...], rgb=[...];
```

B. Pre-Filtering Relevant Objects for Utterance

The aforementioned procedure results in a representation of every object detected in the scene. However, only a small fraction of these objects will typically be relevant to the given referring expression. Given the utterance, “chair in the corner of the room, between white and yellow desks”, the model identifies the following objects as relevant,

Spatial Semantic Scene Description for Relevant Objects

```
scene0592: ... Scene center: [...] ... objs list:
wall, id=8, ctr=[...], size=[...], rgb=[...];
...
armchair, id=15, ctr=[...], size=[...], rgb=[...];
...
chair, id=19, ctr=[...], size=[...], rgb=[...];
```

and the remainder as irrelevant. Simplifying the object list by filtering out irrelevant items (taking into account synonyms and hypernyms) not only reduces processing time and the number of tokens for LLMs, but also facilitates reasoning by reducing content that may potentially be distracting. Such an approach has been shown to improve the efficiency of language grounding [35].

C. Iterative Code Generation and Reasoning

Compositional reasoning that involves arithmetic calculations, which is crucial for spatial reasoning, is well known to be a weakness of Transformers and LLMs [36]. However, most contemporary approaches to 3D referring expression

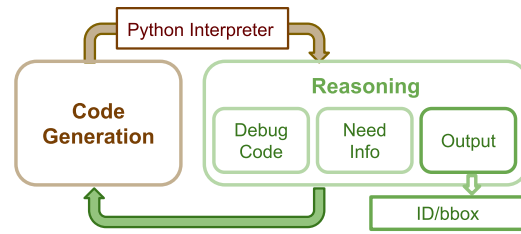


Fig. 3: Illustration of the iterative code generation and reasoning process. After the code generation, execution results from a local Python interpreter is fed back to the LLM for further reasoning. The LLM then proceeds to either 1) fix code errors when encountered, 2) generate additional code to obtain more information, or 3) output the referred objects if it has all the needed information. This process continues until the LLM believes the reasoning to be complete.

resolution rely on a single forward pass of a Transformer to reach the final answer, which can be limiting in terms of reasoning power. In order to avoid this weakness, Transcrib3D equips the LLM with a Python interpreter and directs the LLM to generate code whenever quantitative evaluations are necessary. Transcrib3D then locally executes the generated code using a Python interpreter and appends the output to the original conversation. The resulting prompt is sent back to the LLM to generate the next response. If there is any error in executing the generated code, Transcrib3D also feeds information about the error back to the LLM for re-generation. This process continues until the LLM believes that the reasoning is complete. Figure 3 provides an illustration of this process.

We want to emphasize that our design does not distinguish between code generation and chat reasoning steps. Such a framework simplifies the design, allows for flexible reasoning, and makes it easy for fine-tuning. But in turn, it requires the LLM to have the ability to do both, so language models that are specifically trained for only coding or only chatting will not perform the best within our framework.

D. Principles-Guided Zero-Shot Prompting

Spatial reasoning can require special geometric operations with respect to natural languages that are not familiar with LLMs [36]. For example, when handling left/right spatial relations, which should factor in the observer’s viewpoint, LLMs often naively ground them with smaller/larger values in the x -coordinates. In order to overcome the deficiencies, Transcrib3D employs a suite of general principles to guide LLM reasoning. For example 1) use HSL space rather than RGB space to better match colors with its names; 2) perform vector operations to resolve directional relations; 3) calculate the point-to-plane distance to determine who’s closer to a wall. The full prompts are included in the released code. Although the set of principles work generally well once designed, new environments and new dataset will require certain new set of rules to be adaptive, which makes it labor-intensive and hard to scalable.

Self-Correction for fine-tuning

Assistant:

Based on the calculated metrics, we can see that chair 49 has the smallest distance to the nearest two walls, which means it is most likely to be in the corner of the room. It also has the smallest distance to the nearest white desk and a relatively small distance to the nearest yellow desk, which means it is likely to be between a white desk and a yellow desk. Therefore, chair 49 is the most likely to be the chair described in the description. Now the answer is complete – 'ID':49

User: The correct answer is object 18. Can you double check the information of object 18 and the given prompt and see where you got wrong?

Assistant: Let's revisit the information provided for object 18 and the constraints given in the prompt to identify where the mistake might have occurred.

Object 18 is described as follows: - Type: Chair - Center: [-2.98, -3.31, 0.39] - Size: [0.53, 0.61, 0.81] - RGB: [60, 58, 50]

The constraints from the prompt are: 1. The chair is in the corner of the room. 2. The chair is between a white and a yellow desk.

Upon reviewing the code execution results, it was noted that chair 18 has a relatively small distance to the nearest two walls, indicating it could be in the corner of the room. Additionally, chair 18 has a very small distance to the nearest white desk (0.66) and a reasonable distance to the nearest yellow desk (2.58), which aligns with the constraint of being between a white and a yellow desk.

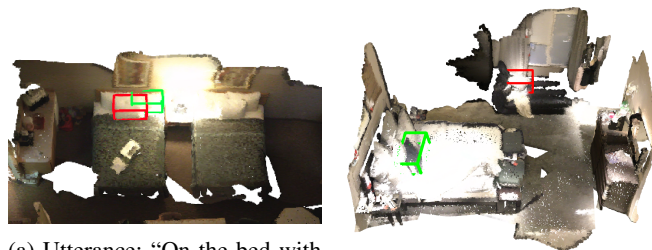
The mistake in the initial analysis was not properly considering the combination of both constraints together. While chair 49 had the smallest distance to the nearest two walls and a very small distance to the nearest white desk, it did not optimally satisfy the condition of being between a white and a yellow desk as well as chair 18 did. Chair 18's proximity to both a white and a yellow desk, along with its corner location, makes it the correct answer according to the given constraints.

Prompt 1: An example of self-reasoned correction for fine-tuning. During self-correction, the LLM realizes its mistake during the initial analysis.

E. Fine-tuning from Self-Reasoned Correction

To overcome the shortcomings of human-designed rule-based system and make the method more adaptive to new environments, We leverage the self-introspection capabilities of large language models (LLMs) to reflect on their mistakes when entering new environments. By collecting these self-reflections and fine-tuning the model based on the insights gained from these experiences, the system continuously learns, improves, and adapts to new challenges and environments. We do so via the following procedure:

- 1) Use the prompt with general principles on the training set for the LLM to generate an initial set of answers with elaborated reasoning process.
- 2) For any incorrect answer, augment the original prompt with the correct object ID and request the LLM to reflect on why the original answer is incorrect (“What went wrong?”).
- 3) Append the reflected reasoning, without the answer, to the original prompt to run the reasoning module again.
- 4) Gather the reasoning processes of the correct examples from the original prompt and the self-reflected examples to produce a dataset for LLM fine-tuning.
- 5) Use supervised fine-tuning (SFT) with quantization [37]



(a) Utterance: “On the bed with the laptop its the pillow by the light.” This query requires understanding three constraints together: “On the bed”, “with the laptop”, and “by the light”. (b) Utterance: “The foremost pillow on the bed of the group of pillows.” This query requires understanding that the pillow needs to be among a group.

Fig. 4: Qualitative comparisons between Transcrib3D (ours, in green) and 3D-VisTA (in red) on the NR3D dataset.

to fine-tune on small-scale language models [38].

Prompt 1 shows an example of the self-reasoned correction step. Note that after fine-tuning, we no longer include general principles in the prompt. In this way, the system not only incorporates the guiding principles, but further improves itself by learning from self-reasoned correction.

IV. EXPERIMENTS

We evaluate the effectiveness of Transcrib3D using the ReferIt3D [1] and ScanRefer [13] benchmarks. ReferIt3D formulates 3D referring expression understanding as a multiple-choice problem: given a set of segmented objects in a 3D scene, identify the unique referent object from a natural language referring expression. ReferIt3D contains two subsets: SR3D, which consists of template-based utterances, and NR3D, which consists of human-sourced free-form utterances. ReferIt3D measures performance in terms of accuracy. Unlike ReferIt3D, ScanRefer does not provide object segmentation information, and instead tasks methods with returning the 3D bounding box of the target object, given the query utterance and a colored point-cloud. ScanRefer measures performance in terms of accuracy conditioned on the intersection-over-union (IoU) of the ground-truth 3D bounding box over the predicted one, with thresholds at 25% (acc@0.25IoU) or 50% (acc@0.5IoU).

A. Grounding Accuracy on ReferIt3D

We test different variations of our method against baselines on the SR3D and NR3D subsets of ReferIt3D. On NR3D, we evaluate our best model, Transcrib3D (GPT4-P), on the full test set, while other variants of Transcrib3D are evaluated on a subset of 500 randomly sampled data points. On SR3D, we evaluate all Transcrib3D models on the same subset of 500 random samples from the test set.¹ Table I presents the results. Figure 4 provides a qualitative comparison between 3D-VisTA [3] and Transcrib3D that highlights the strength of our method with regards to complex reasoning.

¹We use a subset of the test set due to the cost of evaluating all variations of Transcrib3D on the full test set, however we believe that the results would be similar on the full test set due to the templated nature of the utterances.

Method	Nr3D					Sr3D				
	Overall	Easy	Hard	View Dep.	View Ind.	Overall	Easy	Hard	View Dep.	View Ind.
SAT [†] [4]	49.2	56.3	42.4	46.9	50.4	57.9	61.2	50.0	49.2	58.3
BUTD-DETR [†] [5]	54.6	60.7	48.4	46.0	78.0	67.0	68.6	63.2	57.0	67.7
MVT [†] [2]	59.5	67.4	52.7	59.1	60.3	64.5	66.9	58.8	58.4	64.7
ViL3DRel [§] [19]	64.4	70.2	57.4	62.0	64.5	72.8	74.9	67.9	63.8	73.2
3D-VisTA [§] [3]	64.2	72.1	56.7	61.5	65.1	76.4	78.8	71.3	58.9	77.3
Transcrib3D (GPT-3.5-NP)	33.8	42.6	25.0	26.9	36.0	79.3	82.8	70.7	70.6	80.5
Transcrib3D (GPT-3.5-P)	46.6	56.0	37.1	29.9	51.9	80.0	80.8	78.1	82.4	79.7
Transcrib3D (GPT-4-NP)	64.5	71.8	57.1	49.4	71.3	97.4	98.4	94.7	87.3	98.7
Transcrib3D (GPT-4-P)	70.2	79.7	60.3	60.1	75.4	98.4	99.2	96.2	98.2	98.4

TABLE I: Grounding accuracy (%) on Nr3D and Sr3D. [†]denotes results from the official benchmarks while [§]denotes results reported in the respective papers. “P”: “with principles”, “NP”: “no principles”. “Easy” samples has one or none distractors in a scene, while “Hard” samples have two or more. The “View Dep.” (view-dependent) samples contain language descriptions that rely on viewing directions while “View Ind.” (view-independent) do not. All our models are equipped with interactive code generation. Transcrib3D with GPT-4 and general principles surpasses all baselines by a large margin.

B. Grounding Accuracy on ScanRefer

We test the best variant (GPT4-P) of our method with different detection modules against baselines on the ScanRefer benchmark, which does not provide methods with the ground-truth bounding boxes during inference. In addition to comparing to baseline results reported in their respective papers, we also re-run the checkpoints from 3D-VisTA [3], the best baseline method, on the same 500 random samples from the ScanRefer validation set. Table II presents the results. We achieve state-of-the-art performance on ScanRefer with both detected or ground-truth bounding boxes. It is worth noting that the performance gain over the baseline is greater with ground-truth bounding boxes compared to detected ones. This is partly due to the fact that the baseline method is trained with the lower-quality detected bounding boxes, which ironically leads to the ground-truth ones being out of distribution.

C. Effects of Fine-tuning Methods

We study the effects of our proposed approach that involves fine-tuning from self-reasoned corrections. Following Section III-E, we run Transcrib3D with GPT-4 on 500 samples from the NR3D training set and collect both correct and incorrect examples. For the incorrect examples, we let the LLM re-generate the self-corrected reasoning. We use the combination of correct and self-corrected examples to fine-tune the smaller model (i.e., gpt-3.5-turbo) for 3 epochs. The finetuning method used is QLoRA [40] from PEFT library [37]. Sample datasets and configurations are in the released code. We evaluate all models on the same subset of 500 random samples from the NR3D test set. Notably, we remove all rule-based prompts from the fine-tuning data so that the fine-tuned models are not constrained to human-designed rules, but are instead adaptive to new rules learned via self-correction.

Note that a fine-tuned small model with comparable performance to that of a large model on this task would be more

Method	Data	Det.	Overall	
			acc@0.25	acc@0.5
ViL3DRel [19]	Full	PG	47.9	37.7
3D-VisTA [3]	Full	M3D	50.6	45.8
3D-VisTA (re-run)	Full	M3D	50.7	45.9
3D-VisTA (re-run)	Part.	M3D	50.6	44.6
Transcrib3D	Part.	M3D	51.2	44.4
Transcrib3D + Cam	Part.	M3D	51.3	45.5
3D-VisTA (re-run)	Part.	GT	55.6	55.6
Transcrib3D	Part.	GT	62.0	62.0
Transcrib3D + Cam	Part.	GT	64.2	64.2

TABLE II: Grounding accuracy (%) on ScanRefer. “Full”: the full validation set of ScanRefer consisting of 5410 samples; “Part.”: a subset of 500 random samples from the validation set, which is the same for all methods; “Det.”: the 3D object detection module used in the model; “PG” stands for PointGroup [39], while “M3D” stands for Mask3D [12] (where the detection accuracy is 56.7 for PG and 73.7 for M3D on the ScanNet dataset for the mAP@0.5 metric [12]), and “GT” for ground-truth bounding boxes. We test our method and re-run 3D-VisTA on the same subset of 500 samples with the Mask3D detector, GT bounding boxes, and an additional “+ Cam” setting, where camera view information provided by the ScanRefer dataset is also included in the scene transcript. Note that ScanRefer allows the use of all provided data modalities and ranks methods on the same benchmark regardless. The zero-shot nature of our method allows ease use of this extra information.

desirable for local deployment on edge computers or robots, with additional privacy benefits. Additionally, the inference speed of a finetuned small model is significantly faster than that of a bigger model. For example, the reasoning process with a finetuned GPT-3.5 model takes 15 seconds while GPT-4 model may take 1 minute. Further speedup may be realized in the future with small distilled open-sourced models with customized inference infrastructure. Table III presents the

results.

Models	Total	Easy	Hard	Time
GPT-3.5 (zero-shot)	33.8	42.6	24.0	~ 15s
GPT-3.5 (correct-only FT)	60.7	52.8	41.9	~ 15s
GPT-3.5 (self-correct FT)	61.5	54.6	80.0	~ 15s
GPT-4 (zero-shot)	69.4	78.7	60.0	~ 60s

TABLE III: Performance of fine-tuning models on NR3D. Fine-tuned GPT-3.5 models demonstrate a significant improvement in performance when compared to zero-shot models, closely approaching the capabilities of GPT-4. The model fine-tuned on self-corrected examples sees an increase in performance compared to that fine-tuned on only correct examples, particularly for hard queries. Notably, the fine-tuned models are not provided with rule-based prompts during both fine-tuning and inference time, suggesting that implicit decision rules are learned from examples.

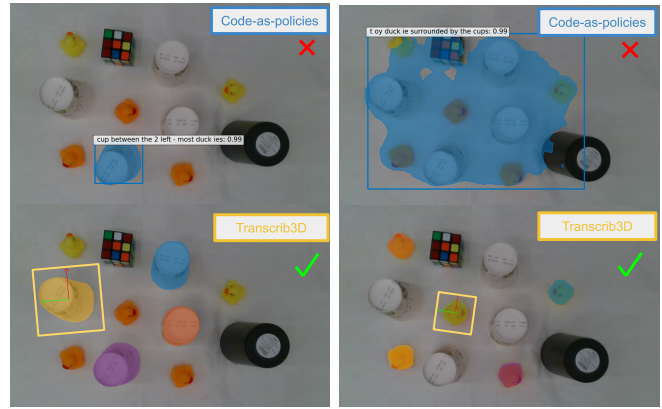
D. Referring Expressions for Robot Manipulation

We demonstrate how Transcrib3D supports a robot’s ability to follow natural-language instructions for pick-and-place manipulation, particularly when complicated referring expressions are involved. Language-guided pick-and-place involves (i) breaking down the language instruction into two referring expressions, one each for the “pick” and “place” identities; (ii) resolving the referring expressions in the context of the robot’s surrounding environment; and (iii) performing the pick-and-place actions. We conduct the pick-and-place task with a Universal Robots UR5 arm equipped with a Robotiq 3-Finger Adaptive Robot Gripper placed in a table-top setting. Figure 2 visualizes the execution of one utterance using our method.

To parse the given instruction, we employ a language model-generated program (LMP) from few-shot prompting as in Code-as-Policies [41], which is instructed to call the `put_first_on_second(arg1, arg2)` function with desired arguments `arg1` and `arg2`. This approach allows free-form texts as input, which is more flexible and natural than template parsing. The following is an example that is used in the prompt for the LMP:

```
'# query: Pick up the orange between the apples \
and place it in the bowl with a banana in it. \
put_first_on_second("orange between the apples", \
"bowl with a banana in it")'
```

The `put_first_on_second(arg1, arg2)` function first composes an exhaustive list of objects in the environment along with their spatial-semantic details. It employs MDETR [42], an open-vocabulary object segmentation method, that takes as input an RGB image and an object category, and outputs the 2D spatial attributes of all objects in the scene of that category. The 2D attributes are then lifted to 3D using depth information from a Realsense RGB-D camera. Repeating the process for all candidate object categories results in a 3D spatial-semantic transcript of all objects in the scene. Transcrib3D then reasons over this transcript along with the referring expression to identify the object in question. Finally, we compute the pick and place



(a) “the cup between the two left-most ducks” (b) “the toy duckie surrounded by the white cups”

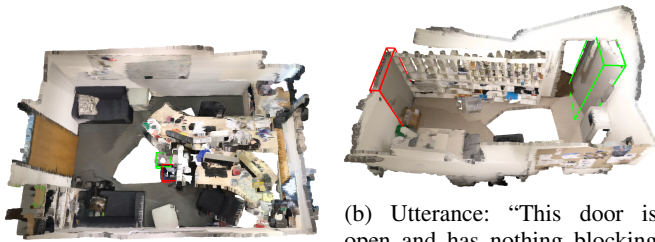
Fig. 5: Qualitative comparison of the grounding performance of (top) CaP+MDETR and (bottom) CaP+Transcrib3D.

poses accordingly to control the robot’s end effector. We provide the pseudo-code below.

```
def put_first_on_second(self, arg1, arg2):
    # obtain the objects list in the environment
    objs = self.env.get_objs()
    pick_id = get_obj_id(objs, arg1) # Referring
    place_id = get_obj_id(objs, arg2) # Referring
    pick_pose, place_pose = self.get_obj_pose(
        objs, pick_id, place_id)
    self.env.step({'pick': pick_pose, \
                  'place': place_pose}) # robot manipulation
```

We apply our method to robot manipulation by integrating Transcrib3D as the perception process in the Code-as-Policies (CaP) [41] framework (“CaP+Transcrib3D”). As a baseline, we compare to the standard implementation of Code-as-Policies, which uses MDETR [42] for perception (“CaP+MDETR”). When grounding an object in the scene, CaP+MDETR assumes that the object can be uniquely identified by MDETR according to its associated noun phrase (e.g., “the duckie”). CaP+MDETR then simply selects the output grounding with the highest score. In order to make the comparison fair, we extend CaP’s prompt so that it is able to use the entire referring expression as the object identifier, which is indeed unique. Otherwise, CaP would perform poorly in our setting, since it has never seen complicated referring expressions in the original prompts.

We evaluate the performance of both methods on five different natural language instructions (each involving two referring expressions). Figure 5 shows a qualitative comparison for one of the test cases. Our results reveal that when assessed at both the referring expression and instruction levels, CaP+Transcrib3D significantly outperforms CaP+MDETR. CaP+Transcrib3D achieves an instruction success rate of 80% (4 out of 5) compared to 20% (1 out of 5) for CaP+MDETR, and correctly resolves 90% of the referring expressions (9 out of 10) compared to 40% (4 out of 10) for CaP+MDETR. This underscores a pronounced benefit in employing Transcrib3D for resolving referring expressions within robot pick-and-place tasks.



(a) Utterance: “The cylinder shaped trash can.” Our model is unable to ground the “cylinder” shape, leading to failure.

(b) Utterance: “This door is open and has nothing blocking it.” This query requires understanding the concept of “open”, which is hard to infer from a bounding box.

Fig. 6: Typical failure cases in Transcrib3D, with ground-truth bounding boxes shown in green and ours in red.

V. DISCUSSION

We acknowledge that Transcrib3D exhibits several limitations. First, compared to full pointcloud, transcription does induce information loss because there is only limited choices we can make when encoding object information into scene description. We include center, size, RGB and orientation (optionally) as our default, which works effectively. However, there are many human referring expressions that require more nuances. We show examples of two such cases in Figure 6. Second, our reliance on existing 3D detectors introduces a constraint: the quality of 3D detection itself. In our experiments, we observed that even state-of-the-art 3D detection methods [12, 43] yield sub-optimal results, highlighting the room for improvement in 3D detection.

However, even with the limitations, our method surpasses all current multi-modality baselines. This achievement leads us to propose two critical insights: firstly, the connection module facilitating interaction between different modalities may not be as effective since our experiments can be regarded as a controlled test that replaces a typical multi-modal cross-attention module with just *text*, and demonstrates that it works better; and secondly, the current reasoning module might be too simplistic to ground the complex logic inherent in natural language. We believe that the observed limitations in current multi-modality methods can largely be attributed to the scarcity of annotated 3D data, which is orders-of-magnitude smaller than its 2D counterpart, primarily due to the higher cost in its collection. We call for more efficient data collection pipelines in 3D and robotics, potentially leveraging generative methods.

VI. CONCLUSION

We introduce Transcrib3D, a simple and effective method for the 3D grounding of natural language referring expressions that requires no training, yet delivers state-of-the-art performance across leading 3D reference resolution benchmarks. The idea of using text as a unifying medium to connect the 3D scene and LLM reasoning not only achieves great results, but also provides critical insights into the bottleneck of current multi-modal methods.

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 *Proc. Int’l. Conf. on Computer Vision (ICCV)*, 2020.

[2] S. Huang, Y. Chen, J. Jia, and L. Wang, “Multi-view transformer for 3D visual grounding,” in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[3] Z. Zhu, X. Ma, Y. Chen, Z. Deng, S. Huang, and Q. Li, “3D-VisTA: Pre-trained transformer for 3D vision and text alignment,” *arXiv preprint arXiv:2308.04352*, 2023.

[4] Z. Yang, S. Zhang, L. Wang, and J. Luo, “SAT: 2D semantics assisted training for 3D visual grounding,” in *Proc. Int’l. Conf. on Computer Vision (ICCV)*, 2021.

[5] A. Jain, N. Gkanatsios, I. Mediratta, and K. Fragkiadaki, “Bottom up top down detection transformers for language grounding in images and point clouds,” *arXiv preprint arXiv:2112.08879*, 2021.

[6] A. Radford *et al.*, “Learning transferable visual models from natural language supervision,” in *Proc. Int’l Conf. on Machine Learning (ICML)*, 2021.

[7] N. Liu, S. Li, Y. Du, A. Torralba, and J. B. Tenenbaum, “Compositional visual generation with composable diffusion models,” in *Proc. Int’l. Conf. on Computer Vision (ICCV)*, 2022.

[8] S. Tong, Z. Liu, Y. Zhai, Y. Ma, Y. LeCun, and S. Xie, “Eyes wide shut? exploring the visual shortcomings of multimodal llms,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2024, pp. 9568–9578.

[9] M. Deitke *et al.*, “Objaverse-XL: A universe of 10M+ 3D objects,” *arXiv preprint arXiv:2307.05663*, 2023.

[10] C. Schuhmann *et al.*, “Laion-5b: An open large-scale dataset for training next generation image-text models,” *Advances in Neural Information Processing Systems*, vol. 35, pp. 25 278–25 294, 2022.

[11] OpenAI, “GPT-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023.

[12] J. Schult *et al.*, “Mask3D: Mask transformer for 3D semantic instance segmentation,” in *Proc. IEEE Int’l Conf. on Robotics and Automation (ICRA)*, 2023.

[13] D. Z. Chen, A. X. Chang, and M. Nießner, “ScanRefer: 3D object localization in RGB-D scans using natural language,” *arXiv preprint arXiv:1912.08830*, 2019.

[14] J. Chen, W. Luo, R. Song, X. Wei, L. Ma, and W. Zhang, “Learning point-language hierarchical alignment for 3D visual grounding,” in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[15] A. Vaswani, N. M. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention is all you need,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[16] J. Hsu, J. Mao, and J. Wu, “NS3D: Neuro-symbolic

- grounding of 3D objects and relations,” in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2023.
- [17] D. Z. Chen, Q. Wu, M. Nießner, and A. X. Chang, “D3Net: A unified speaker-listener architecture for 3D dense captioning and visual grounding,” in *Proc. Int’l. Conf. on Computer Vision (ICCV)*, 2021.
- [18] D. Cai, L. Zhao, J. Zhang, L. Sheng, and D. Xu, “3DJCG: A unified framework for joint dense captioning and visual grounding on 3D point clouds,” in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2022.
- [19] S. Chen, P.-L. Guhur, M. Tapaswi, C. Schmid, and I. Laptev, “Language conditioned spatial relation reasoning for 3D object grounding,” *arXiv preprint arXiv:2211.09646*, 2022.
- [20] T. Gupta and A. Kembhavi, “Visual programming: Compositional visual reasoning without training,” *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 14 953–14 962, 2022.
- [21] M. Ahn *et al.*, “Do as I can, not as I say: Grounding language in robotic affordances,” in *Proc. Conf. on Robot Learning (CoRL)*, 2022.
- [22] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. D. Reid, and N. Sünderhauf, “SayPlan: Grounding large language models using 3D scene graphs for scalable task planning,” *arXiv preprint arXiv:2307.06135*, 2023.
- [23] Y. Hong, H. Zhen, P. Chen, S. Zheng, Y. Du, Z. Chen, and C. Gan, “3D-LLM: Injecting the 3D world into large language models,” *arXiv preprint arXiv:2307.12981*, 2023.
- [24] S. Kobayashi, E. Matsumoto, and V. Sitzmann, “Decomposing NeRF for editing via feature field distillation,” *arXiv preprint arXiv:2205.15585*, 2022.
- [25] C. Qi, L. Yi, H. Su, and L. J. Guibas, “PointNet++: Deep hierarchical feature learning on point sets in a metric space,” in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.
- [26] W. Feng, W. Zhu, T.-J. Fu, V. Jampani, A. R. Akula, X. He, S. Basu, X. E. Wang, and W. Y. Wang, “LayoutGPT: Compositional visual planning and generation with large language models,” *arXiv preprint:2305.15393*, 2023.
- [27] B. Yu, H. Kasaei, and M. Cao, “Co-NavGPT: Multi-robot cooperative visual semantic navigation using large language models,” *arXiv preprint arXiv:2310.07937*, 2023.
- [28] J. Yang, X. Chen, S. Qian, N. Madaan, M. Iyengar, D. F. Fouhey, and J. Chai, “LLM-Grounder: Open-vocabulary 3D visual grounding with large language model as an agent,” *arXiv preprint arXiv:2309.12311*, 2023.
- [29] J. Wei, X. Wang, D. Schuurmans, M. Bosma, E. H. Hsin Chi, F. Xia, Q. Le, and D. Zhou, “Chain of thought prompting elicits reasoning in large language models,” *arXiv preprint arXiv:2201.11903*, 2022.
- [30] S. Yao, D. Yu, J. Zhao, I. Shafran, T. L. Griffiths, Y. Cao, and K. Narasimhan, “Tree of thoughts: Deliberate problem solving with large language models,” *arXiv preprint arXiv:2305.10601*, 2023.
- [31] T. Yoneda, J. Fang, P. Li, H. Zhang, T. Jiang, S. Lin, B. Picker, D. Yunis, H. Mei, and M. R. Walter, “Statler: State-maintaining language models for embodied reasoning,” *arXiv preprint arXiv:2306.17840*, 2023.
- [32] S. Yao, J. Zhao, D. Yu, N. Du, I. Shafran, K. Narasimhan, and Y. Cao, “React: Synergizing reasoning and acting in language models,” *arXiv preprint arXiv:2210.03629*, 2022.
- [33] N. Shinn, F. Cassano, A. Gopinath, K. Narasimhan, and S. Yao, “Reflexion: Language agents with verbal reinforcement learning,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [34] K. Mo, S. Zhu, A. X. Chang, L. Yi, S. Tripathi, L. J. Guibas, and H. Su, “PartNet: A large-scale benchmark for fine-grained and hierarchical part-level 3D object understanding,” in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018.
- [35] S. Patki, A. F. Daniele, M. R. Walter, and T. M. Howard, “Inferring compact representations for efficient natural language understanding of robot instructions,” in *Proc. IEEE Int’l Conf. on Robotics and Automation (ICRA)*, May 2019.
- [36] N. Dziri *et al.*, “Faith and fate: Limits of transformers on compositionality,” *arXiv preprint arXiv:2305.18654*, 2023.
- [37] S. Mangrulkar, S. Gugger, L. Debut, Y. Belkada, S. Paul, and B. Bossan, “Peft: State-of-the-art parameter-efficient fine-tuning methods,” <https://github.com/huggingface/peft>, 2022.
- [38] H. Touvron *et al.*, “Llama 2: Open foundation and fine-tuned chat models,” *arXiv Preprint arXiv:2307.09288*, 2023.
- [39] L. Jiang, H. Zhao, S. Shi, S. Liu, C.-W. Fu, and J. Jia, “PointGroup: Dual-set point grouping for 3D instance segmentation,” in *Proc. IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020.
- [40] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, “Qlora: Efficient finetuning of quantized llms,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [41] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. R. Florence, and A. Zeng, “Code as policies: Language model programs for embodied control,” in *Proc. IEEE Int’l Conf. on Robotics and Automation (ICRA)*, 2022.
- [42] A. Kamath, M. Singh, Y. LeCun, G. Synnaeve, I. Misra, and N. Carion, “MDETR – Modulated detection for end-to-end multi-modal understanding,” in *Proc. Int’l. Conf. on Computer Vision (ICCV)*, 2021.
- [43] Z. Liu, Z. Zhang, Y. Cao, H. Hu, and X. Tong, “Group-free 3D object detection via transformers,” in *Proc. Int’l. Conf. on Computer Vision (ICCV)*, 2021.