Transcribe3D: Grounding LLMs Using Transcribed Information for 3D Referential Reasoning with Self-Corrected Finetuning

Jiading Fang*, Xiangshan Tan*, Shengjie Lin*, Hongyuan Mei, Matthew R. Walter

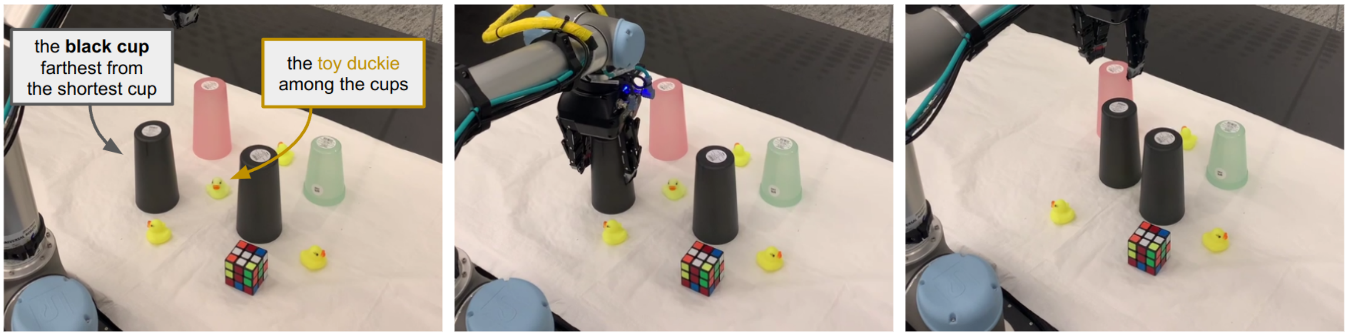

Fig. 1: Transcribe3D enables complex 3D referring in the robotic pick-and-place task. In this example, given the query with challenging referring expressions *"cover the toy duckie among the cups with the black cup farthest from the shortest cup"*, snapshots of the robot movement are shown sequentially from left to right.

***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 as well as to correctly ground free-form language in the presence of distraction and clutter. We propose Transcribe3D, a simple yet effective approach to interpreting 3D referring expressions, which converts 3D scene geometry into a textual representation and takes advantage of the common sense reasoning capability of large language models (LLMs) to make inferences about the objects in the scene and their interactions. We experimentally demonstrate that employing LLMs in this *zero-shot* fashion outperforms contemporary methods. We then improve upon the zero-shot version of Transcribe3D by performing *finetuning from self-correction* in order to generalize to new data. We show preliminary results on the Referit3D dataset with state-of-the-art performance. We also show that our method enables real robots to perform pick-and-place tasks given queries that contain challenging referring expressions.**

I. Introduction

3D referring expression understanding is a task that involves identifying a unique object in a 3D scene given the natural referring expression. The key challenge of this grounding problem lies in the presence of multiple objects similar to the target object that act as distractors. On the standard benchmarks such as Referit3D [1], the strongest methods are still significantly behind human performance. For example, humans achieve approximately 90% accuracy on the human-annotated subset of Referit3D (NR3D), yet the current state-of-the-art method only achieves around 60%. Modern methods typically use a Transformer-based architecture that takes as input (the embeddings of) different modalities (e.g., language, images and point-clouds) and predicts the target object [2–4].

In this work, we take the initiative to investigate the effectiveness of using *text* as the primary modality for interpreting 3D referring expressions, drawing inspiration from the understanding and reasoning capabilities of modern large language models (LLMs) [5–8]. In particular, our LLM-based framework converts spatial and semantic information about 3D scenes—such as the location, orientation, spatial extent, and colors of objects—into text strings that a modern LLM can directly work with. Then the LLM reasons about the given information and identifies the target object.

Precisely, we integrate multiple innovative approaches that are pivotal to the success of our framework, including (i) text-based spatial-semantic 3D scene description; (ii) general principles guided reasoning with interactive code generation; and (iii) finetuning with self-correction from feedback.

First, from the input colored point-cloud, we use off-the-shelf 3D object detection tools like [9] to obtain the objects, each annotated with its type, color, location, orientation and spatial extent (i.e., the 3D bounding box). We compose the information into a list, and feed this text-only scene description along with the query utterance to the LLM.

Second, we propose to use interactive code generation and general principles guided prompting that drastically improve the LLMs' compositional reasoning abilities, which further improve the performance of our framework on a considerable portion of the challenging cases.

Third, in order to scale to new datasets that potentially require different reasoning logic, we propose to automatically incorporate introspection into the model. Our approach handles new reasoning logic by having the language model

*Denotes equal contribution

Toyota Technological Institute at Chicago {fjd, vincenttann, slin, hongyuan, mwalter}@ttic.edu

reflect on its own mistakes when it fails and then generate the correct reasoning for finetuning.

We achieve state-of-the-art results on prominent 3D referring reasoning benchmarks, Referit3D, effectively solving the task with near-human performance. We then deploy our Transcribe3D framework on a real robot manipulator, demonstrating the ability to perform pick-and-place tasks that involve complex free-form commands.

II. METHODOLOGY

In this section, we detail our proposed Transcribe3D framework, which we visualize in Figure 2. We first describe an implementation of the "Referring Model" (Fig. 2 (right)) that uses large language models in a zero-shot fashion with general principles guiding (Section II-D). We then propose the use of introspection as a means of finetuning the reasoning capability of LLMs via self-correction (Section II-E). Sections II-A, II-B and II-C present general techniques that are relevant for both variants of the referring model.

A. Detect and Transcribe 3D Information

Given a colored point-cloud of the scene, we first obtain a semantic segmentation of detected objects using Mask3D [10]. 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 average RGB color. The 3D orientation of an object can also be incorporated using Scan2CAD [11] or PartNet [12]. Transcribe3D compiles the information associated with all detected objects as a list to form an object-centric scene description. Prompt 1 provides an example of such a scene description.

Scene Description

scene0422: objs with quant description based on r-h Cartesian coord sys with x-y-z axes, x-y plane=ground, z-axis=up or down. coords format [x, y, z]. objs list:
trash can,id=8,
ctr=[-3.18,2.68,0.66],size=[0.55,0.56,0.93],rgb=[47,52,45];
recycling bin,id=35,
ctr=[-3.17,2.37,0.49],size=[0.54,0.41,0.75],rgb=[44,46,39];
trash can,id=39,
ctr=[-2.64,2.66,0.54],size=[0.47,0.37,0.78],rgb=[111,130,114];
trash can,id=42,
ctr=[-2.33,2.7,0.52],size=[0.45,0.37,0.76],rgb=[86,89,76];
trash can,id=46,
ctr=[-2.68,-3.05,0.57],size=[0.46,0.39,0.7],rgb=[133,120,107];
picture,id=48,
ctr=[-2.17,-3.09,0.49],size=[0.61,0.17,0.53],rgb=[181,166,138];

Prompt 1: Example scene description with transcribed 3D information.

B. Pre-Filtering of Relevant Objects for the Utterance

The aforementioned procedure results in a representation of every detected object in the scene. However, usually only a small fraction of these objects will be relevant to the given referring expression. For example, scene `scene0006` in ScanNet has an object list produced by II-B (only showing name and id) as

[name=wall,id=0; name=bed,id=1; name=bed,id=2; name=curtain,id=3; name=desk,id=4; name=chair,id=5; name=chair,id=6; name=chair,id=7; name=floor,id=8; name=tv,id=9; name=pillow,id=10; name=pillow,id=11; name=pillow,id=12; name=pillow,id=13; name=nightstand,id=14; name=lamp,id=15; name=telephone,id=16; name=table,id=17; name=pillow,id=18; name=backpack,id=19; name=purse,id=20;]

From the given utterance,

this is a white lamp. it is on a desk.

the following objects are deemed relevant by the model,

[desk (id=4) - Relevant (same name as: 'desk' in utterance), lamp (id=15) - Relevant (same name as: 'lamp' in utterance), table (id=17) - Relevant (synonyms of: 'desk' in utterance)]

while the rest of the items are deemed irrelevant. Simplifying the object list by filtering out irrelevant items 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 [13].

C. Iterative Code Generation, Execution and Debugging

Compositional reasoning that involves arithmetic calculations, which is crucial for spatial reasoning, is well known to be a weakness of LLMs [14]. In order to mitigate this weakness, Transcribe3D equips the LLM with a Python interpreter and directs the LLM to generate code whenever quantitative evaluations are necessary. The generated code is then locally executed using the Python interpreter, of which the outputs are appended to the original conversation. The resulting prompt is sent to the LLM in order to generate the next response. If there is any error in executing the generated code, Transcribe3D feeds information about the error back to the LLM for re-generation. This process continues until the LLM believes that the reasoning is complete. Prompt 2 provides an example of this reasoning procedure.

D. General Principles Guided Prompting

Spatial reasoning can require complex compositional logic that is challenging for autoregressive LLMs [14]. For example, when handling left/right spatial relations, which should be determined according to the observer's viewpoint, LLMs often reason over the concept of left and right simply in terms of smaller/larger values for the x-coordinates of the objects. Related problems arise with behind/in-front-of relations, for which LLMs attempt to reason over the y-coordinates of the objects. In order to overcome these deficiencies, Transcribe3D employs a few general principles to guide LLM reasoning. We find that this set of principles works across 3D referring benchmarks. Prompt 3 provides an example of these principles.

E. Finetuning from Self-Correction with Feedback

Rule-based systems [15] are effective for relatively simple domains that involve a limited set of spatial-semantic

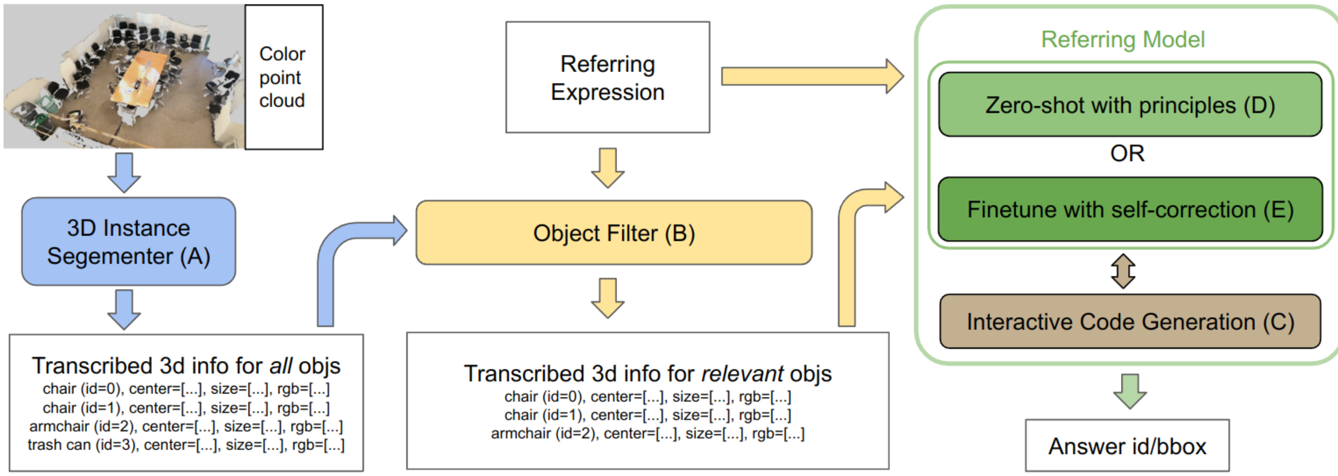

Fig. 2: The overall framework of Transcribe3D.

Interactive Code Generation and Execution

The constraint in the description is 'closest to the foot of the bed'. This is a spatial constraint. We can define a quantitative metric to assess this constraint: the Euclidean distance between the center of each chair and the foot of the bed. The foot of the bed can be determined by the center of the bed and its direction vector. The chair with the smallest distance to the foot of the bed is the one we are looking for.
Let's calculate the distances.

```
import numpy as np
import math
# Define the coordinates of the objects
bed_center = np.array([1.06, -0.75, 0.5])
bed_front_vector = np.array([-1.0, 0.01, 0.0])
bed_size = np.array([2.34, 1.9, 1.02])
chair20_center = np.array([-0.69, -2.11, 0.43])
chair21_center = np.array([1.84, 1.17, 0.44])
# Calculate the coordinates of the bed foot
bed_foot = bed_center + 0.5 * bed_size[0] * \
    bed_front_vector
# Calculate the distances from
# the chairs to the bed foot
dist20 = np.linalg.norm(chair20_center - \
    bed_foot)
dist21 = np.linalg.norm(chair21_center - \
    bed_foot)
print(f"Distance from chair 20 to the \
    foot of the bed: {dist20:.2f}")
print(f"Distance from chair 21 to the \
    foot of the bed: {dist21:.2f}")
```

(Executed locally) Execution result of the above code is:
Distance from chair 20 to the foot of the bed: 1.49
Distance from chair 21 to the foot of the bed: 2.73
Based on the calculated distances, chair 20 is the closest to the foot of the bed. Therefore, the object described in the instruction is chair 20.
Now the answer is complete – 'ID':20

Prompt 2: An example of interactive code generation and execution in the reasoning process.

General Principles for Zero-Shot Prompt

First, you can identify potential objects from the object list based on the category name of the focal object. You should rely on your common sense to comprehensively identify all relevant candidates without writing code. For example, for the category name 'table,' objects such as 'table,' 'desk,' 'end table,' 'coffee table,' and so on from the object list should all be considered as potential candidates.
Then, count(do not write code) and tell me the number of candidate objects. If it is 1, which means only one candidate object, you must directly choose it as answer, then stop your response. For example, if the description is 'the white bathhub on the left of the toilet' and there is only one 'bathhub'-like object in the list, answer it directly, ignore 'white' and 'left of the toilet' constraints.
If there are multiple candidate objects, you can continue. Identify the constraints in the description. There might be multiple constraints to help finding the unique target object from multiple candidate objects. For each constraint, you can define a quantitative metric to assess the degree to which each candidate object satisfies this constraint.
You can write code to calculate the metrics, printing out the metrics of each candidate objects, instead of only print the most possible object id.
Some special tips for some constraints:
- Color(if given). Be lenient with color, because different shades of color mentioned in description can have different RGB values
- Direction relation 'left''right'(if given). To judge obj A on 'left' or 'right' of B, calc vector observer-A and observer-B(both projected to x-y plane). If cross product of vector observer-A and vector observer-B(in this order) has positive z, A on right of B. If z neg, A on left of B. Note that order of cross product matters, put vec observer-A at first. DON'T determine left and right relation by comparing x or y coords.
- Vertical relation such as 'on''above''under'(if given). If obj M has vertical relation with obj N, the x,y coord of ctr of M should be inside the x,y range of obj N, while z of M and z of N should satisfy the corresponding order.

Prompt 3: An example of general principles.

concepts and structured language, but struggle to scale, particularly to open-world domains [16]. Motivated in the high level by the benefits of data-driven alternatives to rule-based methods, we adopt a novel finetuning method for LLMs that enables learning beyond the given set of rules (i.e., the general prompting principles) by reasoning over its own mistakes. In effect, this seeks to endow LLMs with introspection capabilities. The procedure is described as below:

1) Use the prompt with general principles on the training

set for the LLM to generate an initial set of answers with reasoning processes.
2) For any incorrect answer, inform the LLM of the correct object ID, and let it reason about why the original answer is incorrect. After it realizes the mistake, ask it to generate a new reasoning process.
3) Take the reasoning processes of the correct examples, and the re-generated ones of the initially incorrect examples to produce the dataset for LLM finetuning.

Note that after finetuning, we no longer include principles to the prompt. Instead, Transcribe3D relies solely on the examples learned during finetuning. In such a way, the system is not limited by the rules that instruct it, but adapts to exceptions that the LLM makes mistakes on and learns to improve beyond the original rule-based prompt. Prompt 4 shows an example of self-correction.

Self-Correction for Finetuning

Assistant:
Based on the cross products calculated, the chairs on the right of the backpack are chair 2, chair 4, chair 7, chair 9, chair 14, chair 21, and chair 26.
Now, let's consider all the constraints together. The chair we are looking for should be one of the chairs closest to the table, have a color closest to brown, and be on the right of the backpack.
From the results, we can see that chair 14, chair 2, and chair 26 are the closest to the table. Among them, chair 2 has the color closest to brown. And chair 2 and chair 26 are on the right of the backpack. Therefore, the chair we are looking for is chair 2.
Now the answer is complete – 'ID':2
User: The correct answer is 1. Can you double check the information of object 1 and the given prompt and see where you got wrong?
Assistant: I apologize for the confusion. Let's review the information of object 1 and the given prompt.
Object 1 is a chair with its center position at [0.29, 0.94, 0.41], its size in x,y,z direction is [0.59, 0.54, 0.85], and its color in RGB format is [84, 61, 54].
The prompt describes the chair as "brown", "located at the end of the table", and "to the right of the backpack on the floor".
From the previous calculations, we found that chair 1 is not the closest to the table, and its color is not the closest to brown among all chairs. However, it is on the right of the backpack.
It seems that the main issue here is the interpretation of "located at the end of the table". In the previous analysis, we interpreted it as the chair being the closest to the table. However, "at the end of the table" could also mean that the chair is in the direction of the longer side of the table, which is not necessarily the closest.
Let's calculate the direction from the center of the table to each chair, and compare it with the direction of the longer side of the table. The chair with the direction most similar to the direction of the longer side of the table could be the one "located at the end of the table".

Prompt 4: An example of self-correction reasoning for finetuning. LLM misinterprets the concept of "at the end of ...", which leads to the initial incorrect answer. However, when prompted with the correct object ID and told to double check, the LLM realizes the mistake, and learns to correctly utilize the concept of "end" in the reasoning.

III. Experiments

A. Datasets

We evaluate the effectiveness of Transcribe3D using Referit3D [1] as 3D referring expression benchmarks. Referit3D presents 3D referring expression understanding as a multiple-choice problem, assuming that 3D segmented object instances are given. The task is then to identify the uniquely referred object among several instances of the same fine-grained category. Referit3D includes the *SR3D* dataset that contains template-based utterances and the *NR3D* dataset comprised of free-form utterances collected from human annotators. Referit3D measures performance in terms of accuracy.

B. Grounding Accuracy on Referit3D Dataset

We test different variations of our method against baselines on the two subsets of Referit3D, namely SR3D and NR3D, and report the results in Table I.

C. Ablation of Interactive Code Generation on SR3D

We ablate the use of "interactive code generation, execution and debugging" II-C and "general principles guided prompting" II-D on the SR3D dataset, which provides relation type labels for fine-grained analysis. There are 5 different types of relations in SR3D, namely "Horizontal", "Vertical", "Support", "Between" and "Allocentric", distributed roughly as 81%, 4%, 2%, 8% and 5% respectively. Even though "Horizontal" type comprises the majority of the dataset, reasoning on the other four types is equally important and interesting. We use equal number of samples for each type, and show results in Table II.

D. Pick-and-Place Referring for Robotic Manipulations

We incorporate our single-object 3D referring method into pick-and-place robot manipulations. Being arguably the most common robotic manipulation domain, language-based pick-and-place involves (i) breaking down the language query into two referring expressions, corresponding to the "pick" and "place" identities respectively; and (ii) resolving the referring expressions in the context of the robot's surrounding environment. We perform the pick-and-place task with a UR5e robot under the table-top setting. See Figure 1 for an example.

To parse a given command, we employ a language model generated program (LMP) from few-shot prompting as in Code-as-Polices [19], 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. We show an example used in the prompt for the LMP as below.

```
'# 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")'
```

Within the `put_first_on_second(arg1, arg2)` function, an exhaustive list of objects in the environment is first composed. Specifically, we run MDETR [20], an open-vocabulary object segmentation method, with an RGB image

Method	Nr3D					Sr3D				
	Overall	Easy	Hard	View Dep.	View Indep.	Overall	Easy	Hard	View Dep.	View Indep.
SAT† [4]	49.2	56.3	42.4	46.9	50.4	57.9	61.2	50.0	49.2	58.3
BUTD-DETR† [2]	54.6	60.7	48.4	46.0	78.0	67.0	68.6	63.2	57.0	67.7
MVT† [3]	59.5	67.4	52.7	59.1	60.3	64.5	66.9	58.8	58.4	64.7
ViL3DRel§ [17]	64.4	70.2	57.4	**62.0**	64.5	72.8	74.9	67.9	63.8	73.2
3D-VisTA§ [18]	64.2	72.1	56.7	61.5	65.1	76.4	78.8	71.3	58.9	77.3
Transcribe3D (GPT-3.5-NP)	33.8	42.6	25.0	26.9	36.0	79.3	82.8	70.7	70.6	80.5
Transcribe3D (GPT-3.5-P)	46.6	56.0	37.1	29.9	51.9	80.0	80.8	78.1	82.4	79.7
Transcribe3D (GPT-3.5-NP-F-C)	62.6	71.6	53.6	46.3	67.7	97.1	100.0	90.2	94.1	97.6
Transcribe3D (GPT-3.5-NP-F-SC)	63.7	73.8	53.6	53.7	66.8	96.4	99.0	90.2	100.0	95.9
Transcribe3D (GPT-4-NP)	64.5	71.8	57.1	49.4	71.3	97.9	97.0	**100.0**	88.2	**99.2**
Transcribe3D (GPT-4-P)	**69.4**	**78.7**	**60.0**	55.2	**73.8**	**98.6**	**100.0**	95.1	**100.0**	99.1

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”, “F”: “finetuning”, “C”: “correct cases only”, “SC”: “correct and self-correction cases”. Our finetuned models are trained on 500 reasoning data samples from GPT-4 on NR3D. Our models are evaluated on 281 samples on the test set of NR3D and 140 examples on SR3D due to limited computational budget. The results show that Transcribe3D with GPT-4 and general principles surpasses all baselines by a large margin. Finetuned GPT-3.5 models have close performance compared to GPT-4 even without general principles, and self-correction brings noticeable improvement.

Type	Horiz.	Vert.	Supp.	Betw.	Allocent.
No-Code & NP	93.3	93.3	83.3	96.7	66.7
Code & NP	96.7	93.3	90.0	96.7	70.0
Code & P	100.0	90.0	100.0	96.7	90.0

TABLE II: Ablation of “interactive code generation” and “general principles guided prompting” on SR3D. “Code”: “interactive code generation”, “NP”: “no principles”, “P”: “with principles”. Results show that “Code” and “Principles” help in the challenging cases like “Allocentric” and “Support“.

and the list of candidate objects as input, obtaining a 2D spatial-semantic description of the scene. By incorporating depth information from a RealSense camera, the description is lifted to 3D, and then used as input for our referring module to identify the objects in question. Finally, pick and place poses are computed accordingly to manipulate the robot’s end effector. We provide the pseudocode as 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
```

IV. Related Works

A. The 3D Referring Task and Methods

The 3D referring task was initiated by the benchmark of Referit3D, which has the *SR3D* subset that contains template-based utterances and the *NR3D* subset for free-form utterances collected from human annotators.

Significant attention has been paid of late to the problem of interpreting 3D referring expressions. On Referit3D, the highest-performing methods relevant to our work are MVT [3] (NR3D and SR3D), BUTD-DETR [2] (NR3D and SR3D), SAT [4] (NR3D) and NS3D [21] (SR3D).

MVT, BUTD-DETR and SAT all propose to merge different input modalities into a combined embedding using Transformers-like architectures [22] in an end-to-end fashion. MVT projects 3D information into 2D to achieve a better feature encoding. BUTD-DETR finetunes detected 3D bounding boxes within the Transformer. SAT uses 2D semantics in training to facilitate learning a mapping between the query and their 3D groundings. Different from these methods, NS3D proposes a neural-symbolic way that leverages language-to-code model to generate programs, where each module is represented by neural networks.

B. Groundings for Large Language Models

Large language models (LLMs) trained on Internet-scale text data have shown dominant performance across various NLP tasks [5, 6, 8]. However, LLMs have to be grounded so as to answer questions about the real world for robotic applications. SayCan [23] and SayPlan [24] use predicted robotic affordance by separate modules or scene graph as the grounding. Other works train multi-modality models that directly incorporate 3D representations into the token library [18, 25]. 3D-LLM [25] incorporates distilled 3D features [26] with language tokens. 3D-Vista [18] does multi-modal fusion of language and PointNet++ features [27] with self-supervised masked encoding training. However, our text-only approach outperforms the best multi-modality approach on the 3D Referring task by a large margin [18]. Although we believe the multi-modality approach is promising in the future, in this paper, we would like to push the text-only approach so as to lay out the foundation to answering the scientific question of what we could gain by leveraging multi-modality and when language is insufficient for grounding.

C. Large Language Models Finetuning

There is a large effort in the LLM community to finetune the pre-trained language models in order to improve performance on specific target tasks (e.g., conversations, coding) or to align their outputs with human values [28, 29]. However, human preference data are collected in the form of sparse binary signals, so techniques like RLHF [30, 31] or DPO [32] are introduced to facilitate the training. On the other hand, when external signals are absent, there are works that propose to use self-evaluation by LLMs as supervision [33].

In this paper, we perform finetuning in a manner that differs from both of these approaches. We have labels from the training dataset and so we do not rely on self-evaluation, but we do not feedback using binary right-or-wrong signals. Instead, if the inference is incorrect, we provide the LLM with the correct object ID, and instruct it to identify its own reasoning mistakes as a form of introspection, and then to re-generate a full correct reasoning process. We take the reasoning process of both correct examples and self-corrected ones together for finetuning. In this way, we provide denser signal for finetuning and allow the finetuned model to adapt to new data with potentially different reasoning logic.

V. Limitations and Future Works

We acknowledge several limitations to our approach. First of all, since we rely on off-the-shelf 3D detectors, the quality of the 3D detection is a bottleneck. During our tests, even some of the state-of-the-art 3D detection models [9, 10] are producing results less than desired. We think 3D detection methods need to be improved in order to enable future robotic foundation models that can be grounded in 3D. Secondly, currently our scene description is object-centric. While object-level information is sufficient for many 3D-related tasks, there are many cases that do require a more fine-grained resolution of objects, e.g. color of a non-uniform colored object, referring expression related to the shape of an object. Thirdly, we do manually specify the kind of information we want to extract from the 3D detection, i.e. center, size, orientation etc. However, such design may be non-optimal, an adaptive feature-selection strategy or more expressive vector representation may be desired.

Overall, while we showed that text-as-grounding is a surprisingly simple and effective approach to serve as the bridge between 3D scenes and LLM, our main purpose is not to go against the multi-modality approach of building 3D robotic foundation models, but instead we want to explore the limit of language-only approach and hope it could serve as the basis for scientifically understanding what can be gained with multi-modality 3D robotic foundation models.

VI. Conclusion

In this paper, we propose Transcribe3D, a simple yet effective approach to the 3D grounding of natural language referring expressions. We show that transcribing relevant spatial-semantic 3D scene information into a purely textual form can serve as a bridge between 3D scenes and large language models. We show preliminary results on the Referit3D dataset with state-of-the-art performance. We introduce crucial techniques including object filtering, interactive code generation and execution, general principles guided prompting and importantly, a finetuning method that endows LLMs with the capability for introspection that allows them to learn improved reasoning from the self-correction of its own mistakes based on feedback. The finetuning method not only improves performance, but also allows the model to adapt to unseen datasets that potentially require different reasoning logic.

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 *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[2] 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.

[3] S. Huang, Y. Chen, J. Jia, and L. Wang, "Multi-view transformer for 3D visual grounding," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[4] Z. Yang, S. Zhang, L. Wang, and J. Luo, "SAT: 2D semantics assisted training for 3D visual grounding," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[5] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "LLaMA: Open and efficient foundation language models," *arXiv Preprint arXiv:2302.13971*, 2023.

[6] H. Touvron, L. Martin, K. R. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. M. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. S. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. M. Kloumann, A. V. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," *arXiv Preprint arXiv:2307.09288*, 2023.

[7] OpenAI, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[8] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. J. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *arXiv Preprint arXiv:2005.14165*, 2020.

[9] Z. Liu, Z. Zhang, Y. Cao, H. Hu, and X. Tong, "Group-free 3D object detection via transformers," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021, pp. 2929–2938.

[10] J. Schult, F. Engelmann, A. Hermans, O. Litany, S. Tang, and B. Leibe, "Mask3D: Mask transformer for 3D semantic instance segmentation," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2023.

[11] A. Avetisyan, M. Dahnert, A. Dai, M. Savva, A. X. Chang, and M. Nießner, "Scan2CAD: Learning CAD model alignment in RGB-D scans," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[12] 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 *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[13] S. Patki, A. F. Daniele, M. R. Walter, and T. M. Howard, "Inferring compact representations for efficient natural language understanding of robot instructions," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2019, pp. 6926–6933.

[14] N. Dziri, X. Lu, M. Sclar, X. L. Li, L. Jian, B. Y. Lin, P. West, C. Bhagavatula, R. L. Bras, J. D. Hwang, S. Sanyal, S. Welleck, X. Ren, A. Ettinger, Z. Harchaoui, and Y. Choi, "Faith and fate: Limits of transformers on compositionality," *arXiv preprint arXiv:2305.18654*, 2023.

[15] T. Winograd, "Procedures as a representation for data in a computer program for understanding natural language," Ph.D. dissertation, Massachusetts Institute of Technology, 1971.

[16] S. Tellex, N. Gopalan, H. Kress-Gazit, and C. Matuszek, "Robots that use language," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 25–55, 2020.

[17] 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.

[18] 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.

[19] 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 *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2022, pp. 9493–9500.

[20] A. Kamath, M. Singh, Y. LeCun, G. Synnaeve, I. Misra, and N. Carion, "MDETR - Modulated detection for end-to-end multi-modal understanding," in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2021.

[21] J. Hsu, J. Mao, and J. Wu, "NS3D: Neuro-symbolic grounding of 3D objects and relations," in *Proceedings*

of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2023.

[22] 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.

[23] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. M. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan, "Do as I can, not as I say: Grounding language in robotic affordances," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2022.

[24] 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.

[25] 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*, vol. abs/2307.12981, 2023.

[26] S. Kobayashi, E. Matsumoto, and V. Sitzmann, "Decomposing NeRF for editing via feature field distillation," *arXiv preprint arXiv:2205.15585*, 2022.

[27] 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.

[28] J. E. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, and W. Chen, "LoRA: Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021.

[29] A. Glaese, N. McAleese, M. Trkebacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger, M. Chadwick, P. Thacker, L. Campbell-Gillingham, J. Uesato, P.-S. Huang, R. Comanescu, F. Yang, A. See, S. Dathathri, R. Greig, C. Chen, D. Fritz, J. S. Elias, R. Green, S. Mokr'a, N. Fernando, B. Wu, R. Foley, S. Young, I. Gabriel, W. S. Isaac, J. F. J. Mellor, D. Hassabis, K. Kavukcuoglu, L. A. Hendricks, and G. Irving, "Improving alignment of dialogue agents via targeted human judgements," *arXiv preprint arXiv:2209.14375*, 2022.

[30] Y. Bai, A. Jones, K. Ndousse, A. Askell, A. Chen, N. DasSarma, D. Drain, S. Fort, D. Ganguli, T. J. Henighan, N. Joseph, S. Kadavath, J. Kernion, T. Conerly, S. El-Showk, N. Elhage, Z. Hatfield-Dodds, D. Hernandez, T. Hume, S. Johnston, S. Kravec, L. Lovitt, N. Nanda, C. Olsson, D. Amodei, T. B. Brown, J. Clark, S. McCandlish, C. Olah, B. Mann, and J. Kaplan, "Training a helpful and harmless assistant with reinforcement learning from human feedback," *arXiv preprint arXiv:2204.05862*, 2022.

[31] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, and G. Irving, "Fine-tuning language models from human preferences," *arXiv preprint arXiv:1909.08593*, 2019.

[32] R. Rafailov, A. Sharma, E. Mitchell, S. Ermon, C. D. Manning, and C. Finn, "Direct preference optimization: Your language model is secretly a reward model," *arXiv preprint arXiv:2305.18290*, 2023.

[33] N. Shinn, F. Cassano, B. Labash, A. Gopinath, K. Narasimhan, and S. Yao, "Reflexion: Language agents with verbal reinforcement learning," *arXiv preprint arXiv:2303.11366*, 2023.