

NS3D: Neuro-Symbolic Grounding of 3D Objects and Relations

Joy Hsu
Stanford University
joycj@stanford.edu

Jiayuan Mao
Massachusetts Institute of Technology
jiayuanm@mit.edu

Jiajun Wu
Stanford University
jiajunwu@cs.stanford.edu

Abstract

Grounding object properties and relations in 3D scenes is a prerequisite for a wide range of artificial intelligence tasks, such as visually grounded dialogues and embodied manipulation. However, the variability of the 3D domain induces two fundamental challenges: 1) the expense of labeling and 2) the complexity of 3D grounded language. Hence, essential desiderata for models are to be data-efficient, generalize to different data distributions and tasks with unseen semantic forms, as well as ground complex language semantics (e.g., view-point anchoring and multi-object reference). To address these challenges, we propose NS3D, a neuro-symbolic framework for 3D grounding. NS3D translates language into programs with hierarchical structures by leveraging large language-to-code models. Different functional modules in the programs are implemented as neural networks. Notably, NS3D extends prior neuro-symbolic visual reasoning methods by introducing functional modules that effectively reason about high-arity relations (i.e., relations among more than two objects), key in disambiguating objects in complex 3D scenes. Modular and compositional architecture enables NS3D to achieve state-of-the-art results on the ReferIt3D view-dependence task, a 3D referring expression comprehension benchmark. Importantly, NS3D shows significantly improved performance on settings of data-efficiency and generalization, and demonstrate zero-shot transfer to an unseen 3D question-answering task.

1. Introduction

Interacting with the physical world requires 3D visual understanding; it entails the ability to interpret 3D objects and relations among multiple entities, as well as reason about 3D instances in a scene from language expressions. However, due to the variability of the 3D domain, there are two prevalent challenges: the expense of annotating 3D labels and the complexity of 3D grounded language. In this paper, we tackle these two challenges on a specific task of 3D scene understanding, the referring expression comprehension (3D-REC) task. As shown in Figure 1, in a 3D-REC task, the input contains a sentence and a 3D scene,

Complex 3D Grounding in 3D-REC

Instruction: Looking at the front of the copier, pick the **printer** that is to the right of the copier.



Instruction: Facing the front of the cabinet, choose the **lamp** that is on the left of it.

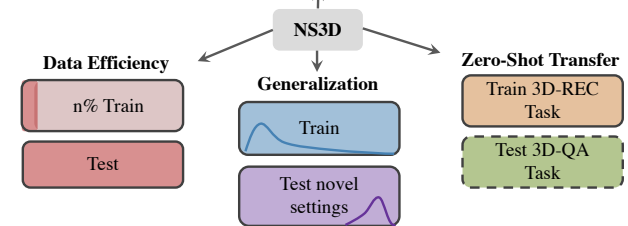


Figure 1. NS3D achieves grounding of 3D objects and relations in complex scenes, while showing state-of-the-art results in data efficiency, generalization, and zero-shot transfer.

usually given as a collection of object point clouds; the goal is to identify the correct referred object in the scene. The task is challenging: obtaining high-quality annotations for such tasks is expensive; the referring expressions often require reasoning about multiple objects, such as anchoring speaker viewpoints (i.e., facing X , select the object Y behind Z) and utilizing multiple objects in the scene as reference points.

Many prior works have studied end-to-end methods to tackle this problem [1, 2, 16–18, 20, 30, 37, 39, 40], jointly attending over features from language and point clouds. These methods report strong performance, but generally require large amounts of data to train and are prone to dataset biases, such as object co-occurrences. Meanwhile, the learned 3D representations cannot be directly transferred to related downstream tasks, such as 3D question answering. In addition, most prior works in 3D grounding are based on Transformers [33], which reduce the set of realizable functions to a subset of reasoning tasks with binary relations [26, 33]. This has limited their ability to resolve complex 3D grounded languages, empirically leading to a noticeable performance

drop when the language contains view-dependent relations.

To this end, we propose NS3D as a powerful neuro-symbolic approach to solve 3D visual reasoning tasks, with more faithful grounding of 3D objects and relations. NS3D first parses the referring expression from the free language form to a neuro-symbolic program form. We introduce the use of Codex [11], a large language-to-code model, for semantic parsing with a small number of prompting examples, leading to perfect identification of entities and program structures. Such program structures decompose each referring expression into a set of functional modules that are hierarchically chained together. Functional modules can perform an object-level grounding step, such as selecting the *bathroom vanity* from the input point clouds, and a relational grounding step, such as finding objects that are *behind* another reference object. This functional composition strategy can be easily extended to more complex functions that require multiple objects, such as view-dependent relation grounding. In NS3D, functional modules are implemented as different neural networks that take object features of the corresponding arity: e.g., object-level grounding modules take per-object features, while relation grounding modules take a set of vector encodings for each pair of objects. Importantly, NS3D extends prior neuro-symbolic approaches for visual reasoning [24] by introducing modules that execute high-arity programs, such as those for relation grounding over multiple objects, especially ubiquitous in the 3D domain.

The combination of compositional structures and modular neural networks fulfills many desiderata for 3D visual reasoning (see Figure 1). First, specializing neural modules for relations that involve multiple objects improves performance, particularly in resolving complex view-dependent referring expressions. Our approach is noticeably simpler and more effective than existing models that solve this task by fusing multiple view representations [18]. Second, the disentangled grounding of objects and relations brings significantly improved data efficiency. Third, by following symbolic structures to compose functional modules, NS3D generalizes better to scenarios with unseen object co-occurrences and scene types. Fourth, the compositional nature of the functional structures and the flexibility of our Codex-based parser enables NS3D to zero-shot generalize to novel reasoning tasks, such as 3D visual question answering (3D-QA). Furthermore, as a byproduct of our modular approach, NS3D enables better interpretability, allowing attribution to where visual grounding fails and succeeds; we show in ablations that NS3D learns almost perfect relation grounding.

We validate NS3D on the ReferIt3D benchmark, which evaluates referring expression comprehension in 3D scenes, and requires fine-grained object-centric and multi-object relation grounding [2]. We report state-of-the-art view-dependent accuracy and comparable overall accuracy to top-performing methods. We also present results on data ef-

iciency and generalization to unseen object co-occurrences and new scenes, with our neuro-symbolic method outperforming all prior work by a large margin. Finally, we show NS3D’s ability to zero-shot transfer from the 3D reference task to a new 3D visual question answering task, achieving strong performance without any data in this novel setup.

To summarize, the contribution of this paper is three-fold: 1) We propose a neuro-symbolic method to ground 3D objects and relations that integrates the power of large language-to-code models and modular neural networks. 2) We introduce a neural program executor that reasons about high-arity relations as a principled solution to view-point anchoring and multi-object reference. 3) We show state-of-the-art view-dependent grounding results in 3D-REC tasks, high accuracy in data-efficient settings (a 24.5 percent point gain from prior work with 1.5% of data), significant improvements in generalization to different data distributions, and ability to zero-shot transfer to an unseen 3D-QA task.

2. Related Work

3D grounding. Many prior works that tackle the 3D-REC task employ end-to-end approaches that jointly attend over language and point clouds [7–10, 23], commonly leveraging a Transformer architecture [33]. These methods can be broadly categorized into two types: object-centric ones, and ones that model the full 3D scene. Most works based on full 3D scene modeling use a detection module to create object proposals. For example, Text-guided Graph Neural Network [17] conducts instance segmentation on the full scene to create candidate objects as input to a graph neural network [32]; InstanceRefer [39] selects instance candidates from the panoptic segmentation of point clouds; 3DVG-Transformer [40] uses outputs from an object proposal generation module to fully leverage contextual clues for cross-modal proposal disambiguation. The best performing work in this category, BUTD-DETR [20], uses box proposals from a pretrained detector and scene features from the full 3D scene to decode objects with a detection head. The Multi-View Transformer [18] separately models the scene by projecting the 3D scene to a multi-view space, to eliminate dependence on specific views and learn robust representations.

By contrast, object-centric models perform reasoning over an input set of object point clouds. ReferIt3DNet [2] utilizes a graph convolutional network with input objects as nodes of the graph. 3DRefTransformer [1], LanguageRefer [30] TransRefer [16], and SAT [37] are Transformer-based methods that operate on language and 3D object point clouds. 3DRefTransformer [1] is an end-to-end Transformer model that incorporates an object pairwise spatial relation loss. LanguageRefer [30] uses a Transformer architecture over bounding box embeddings and language embedding from DistilBert [31]. TransRefer [16] utilizes a Transformer-based network to extract entity-and-relation-aware represen-

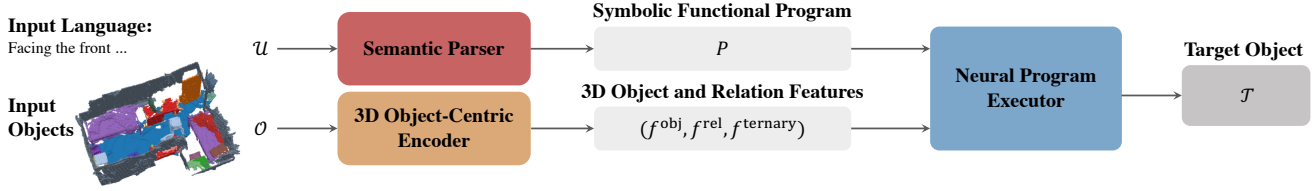


Figure 2. NS3D is composed of three main components. a) A *semantic parser* parses the input language into a symbolic program. b) A *3D object-centric encoder* takes input objects and learns object, relation, and ternary relation features. c) A *neural program executor* executes the symbolic program with the learned features to retrieve the target referred object.

tations. SAT [37] leverages 2D image semantics with a multi-modal Transformer for joint representation learning. NS3D lives in this category of methods, focusing on grounding objects and relations over an object-centric representation. In contrast to prior works, NS3D enables strong data efficiency, generalization, and zero-shot transfer to novel tasks, while not restricted to functions that Transformers can realize or constrained by the need for additional 2D data.

Neuro-symbolic visual reasoning methods. Neuro-symbolic methods have shown strong data efficiency and generalization capability in the 2D visual reasoning domain, from visual question answering to image caption retrieval [3, 4, 12, 15, 19, 21, 22, 25, 35, 36, 38], with the Neuro-Symbolic Concept Learner [24] as a representative work. However, the 3D domain poses additional challenges, such as more complex, high-arity programs, required for anchoring speaker view and using multiple reference objects to resolve a referring expression. NS3D builds on a 3D scene-graph like representation [5, 34]. It retains all the benefits of existing neuro-symbolic visual reasoning models, while extending them to this challenging 3D domain. In addition, NS3D sheds new lights on a broad criticism of prior neuro-symbolic methods on their use of a predefined grammar or trained parser [14, 24]: NS3D leverages large language-to-code models for semantic parsing [11].

3. NS3D

In this section, we describe NS3D applied to the task of referring expression comprehension (3D-REC). We set up the task following ReferIt3D [2]: given a set of M objects in the scene $\mathcal{O} = \{O_1, \dots, O_M\}$, where each object is represented as an RGB-colored point cloud of N points $O_i \in \mathbb{R}^{N \times 6}$, and given an utterance \mathcal{U} , the goal is to predict the target referred object $\mathcal{T} \in \mathcal{O}$. Due to the existence of many distractor objects in \mathcal{O} , it is crucial to parse the full referring expression to select the correct target.

NS3D is a neuro-symbolic approach that combines grammatical functional structures and modular neural networks. It consists of three main components (see Figure 2). The first is a *semantic parser* that parses the input language \mathcal{U} into a symbolic program P that resembles a hierarchical reasoning process underlying \mathcal{U} (Section 3.1). The second is a *3D feature encoder* that extracts an object-centric representa-

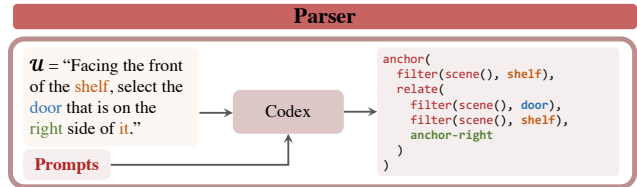


Figure 3. The NS3D semantic parser leverages Codex to parse input language into symbolic programs.

tion f from the input point clouds of objects \mathcal{O} (Section 3.2). The third is a *neural network-based program executor* that takes the symbolic program and the learned object-centric representation, and returns the target object \mathcal{T} (Section 3.3).

3.1. Semantic parser

The goal of the NS3D parser is to parse utterances \mathcal{U} into a symbolic program P that resembles the underlying reasoning process of \mathcal{U} . The program has a hierarchy of primitive operations defined in a minimal but powerful domain-specific language (DSL) for 3D visual reasoning tasks. Each operation is composed of a function name (e.g., *anchor* or *filter*), and arguments (e.g., *shelf*, *door*, *right*). A key feature of such programs is that the output of one operation can be the input (argument) to another operation, as shown in Figure 3. Informally, the *anchor* program grounds the viewpoint, the *filter* program takes all the input objects in the full *scene* and outputs those that are of the specified category, and the *relate* program returns objects that satisfy the given relationship constraint. We include a more formal definition of these operations and the DSL in the supplementary material.

Parsing the input language into this hierarchical program allows NS3D to disentangle the learning of different functional modules that perform object-level or relational grounding and reasoning. These neural programs can be trained and combined in different ways.

Semantic parsing with Codex. In contrast to most existing neuro-symbolic reasoning frameworks, e.g., [24], instead of using a pretrained or jointly-trained semantic parser, we introduce the use of large language-to-code models for parsing. Specifically, we use Codex [6, 11] with the Synchronesh framework [27]. By specifying only a small number of examples of language input and expected programs, we gain perfect parsing capabilities across unseen categories and re-

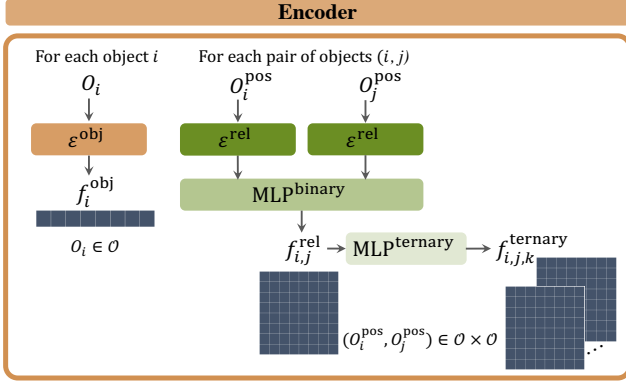


Figure 4. The NS3D object-centric encoder learns object, relation, and ternary relation features from input object point clouds.

lations in the ReferIt3D task. Synchromesh constrains the output of Codex to be a valid, executable program, adhering to the syntactic rules of the DSL.

Using Codex as our semantic parser has two major advantages. First, it only requires a small number of example programs to achieve strong parsing accuracy, compared to defining rules for semantic parsers or training from scratch. This enables us to easily generalize to new DSLs and tasks, such as recombining the learned functional modules in a completely new way to answer visually grounded questions. Second, compared to existing works that assume a given set of visual concepts (categories and relations) [19, 24], Codex can automatically identify unseen concepts from language through its built-in knowledge, even if they never appear in the prompting examples. In our experiments, we show that Codex outperforms a T5-based parser [29] finetuned on the same set of prompting examples.

3.2. 3D object-centric encoder

NS3D’s 3D encoder generates object-centric and relational features for each scene in a latent space for 3D grounding (see Figure 4). Recall that the input to the encoder is a collection of object point clouds $\mathcal{O} = \{O_1, O_2, \dots, O_M\}$, where M is the number of objects. For each 3D object point cloud, the encoder first extracts an object feature vector through a PointNet++ backbone \mathcal{E}^{obj} [28]. \mathcal{E}^{obj} takes as input every object $O_i \in \mathbb{R}^{1024 \times 6}$, representing the RGB color of each point and their XYZ location in the Cartesian space, and outputs encoded features,

$$f_i^{\text{obj}} = \mathcal{E}^{\text{obj}}(O_i), \forall O_i \in \mathcal{O}.$$

Next, for each pair of 3D objects (O_i, O_j) , NS3D uses a separate encoder to extract their relational feature vector. The relation encoders are designed not to share weights with the object encoders, allowing them to learn semantically relevant features for relational reasoning. Specifically, NS3D first encodes each object with a different PointNet++ network \mathcal{E}^{rel} . \mathcal{E}^{rel} takes in only the XYZ positions of object point

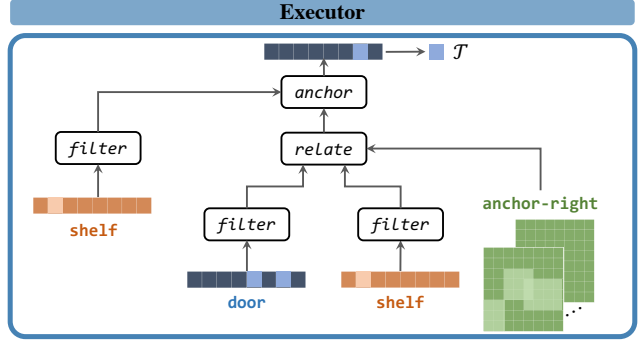


Figure 5. The NS3D neural program executor executes the symbolic program recursively with the learned 3D features, and returns the target referred object \mathcal{T} .

clouds $O_i^{\text{pos}} \in \mathbb{R}^{1024 \times 3}$, as we are interested in modeling the spatial relations between objects rather than object types. Next, NS3D concatenates the per-object encoding for O_i and O_j and applies a small 2-layer multi-layer perceptron $\text{MLP}^{\text{binary}}$ to extract relational features for the object pair:

$$f_{i,j}^{\text{rel}} = \text{MLP}^{\text{binary}} \left(\text{concat} \left(\mathcal{E}^{\text{rel}}(O_i^{\text{pos}}), \mathcal{E}^{\text{rel}}(O_j^{\text{pos}}) \right) \right),$$

where concat denotes the concatenation operation for two vectors. In our design, \mathcal{E}^{rel} is a shallower network that uses a sparser amount of samples than the object feature encoder \mathcal{E}^{obj} , which requires more fine-grained encoding of point clouds to classify categories.

We also model ternary relations as seen in ReferIt3D (e.g., a vector embedding for each triple of objects (O_i, O_j, O_k)). Specifically, we propose using the encoder to also extract a ternary feature $f_{i,j,k}^{\text{ternary}}$. As both the binary-relation features and ternary-relation features focus on encoding spatial relationships among objects, NS3D shares the underlying PointNet encoder for them. This reduces the time and memory cost for high-arity inference. Mathematically,

$$g_{i,j}^{\text{ternary}} = \text{MLP}^{\text{ternary}} \left(f_{i,j}^{\text{rel}} \right),$$

$$f_{i,j,k}^{\text{ternary}} = \text{concat} \left(g_{i,j}^{\text{ternary}}, g_{j,k}^{\text{ternary}}, g_{i,k}^{\text{ternary}} \right),$$

where $\text{MLP}^{\text{ternary}}$ is another 2-layer multi-layer perceptron that shares the same architecture as $\text{MLP}^{\text{binary}}$.

3.3. Neural program executor

The neural program executor takes the parsed program P and the learned representation $(f_i^{\text{obj}}, f_i^{\text{rel}}, f_i^{\text{ternary}})$ for each scene as input, and executes the program based on the 3D representation, returning the target object being referred to. At a high-level, NS3D follows the hierarchical structure in P and executes the program recursively (see Figure 5). Here, we describe the neural implementation for each operation.

The key representation we will be using during program execution is the object score vector, which is a vector of length M (the number of objects in the scene), indicating

whether an object has been selected or not. For example, semantically, the operation *filter* takes an input set of objects being selected and an object category, and outputs a subset of input objects belonging to the specified category. Both the input and output of the *filter* operation will be represented as such object score vectors. For numerical stability, such scores are represented in the log space. One interpretation is that each entry v_i in a score vector represents the log probability of object O_i being selected.

scene() $\rightarrow y$: the scene operation returns a object score vector representing “all objects in the scene.” Recall that the values are in the log space; therefore, $y_i = 0$, for all $i \in \{1, 2, \dots, M\}$.

filter(x, c) $\rightarrow y$: the filter operation takes an input object score vector x , an object category c , and returns a new object score vector, selecting objects that are in x and belongs to category c . Therefore, we first compute $prob_i^c = \text{MLP}^c(f_i^{\text{obj}})$, which is a score for object i belonging to category c , where MLP^c is a mapping (specialized for c) from the dimension of f_i^{obj} to dimension 1. Next, we merge it with the input object score vector x . Overall,

$$y_i = \min(x_i, prob_i^c) = \min\left(x_i, \text{MLP}^c\left(f_i^{\text{obj}}\right)\right).$$

relate(x^t, x^r, rel) $\rightarrow y$: the *relate* program takes as input two sets of objects, target objects x^t and reference objects x^r , as well as a relational concept *rel*, and outputs target objects that satisfy the specified relation. As an example, the expression “the chair beside the shelf” will be parsed into the program *relate*(*filter*(*chair*), *filter*(*shelf*), *beside*)*. In this case, x^t will be the *filter* result for *chair*, while x^r will be the *filter* result for *shelf*. The *relate* operation classifies whether each pair of objects satisfy the relation *rel*, and selects the objects in x^t that have relation *rel* with objects in x^r :

$$prob_{i,j}^{\text{rel}} = \text{MLP}^{\text{rel}}(f_{i,j}^{\text{rel}})$$

$$y_i = \min\left(x_i^t, \sum_j sx(x^r)_j \cdot prob_{i,j}^{\text{rel}}\right),$$

where MLP^{rel} is a linear layer with scalar output and specialized for concept *rel*, and *sx* is the softmax function applied to x^r . The \sum_j operator can be interpreted as a “soft” selection of the j^* -th row in the relation matrix $prob^{\text{rel}}$, where $j^* = \arg \max x^r$, the index of the referred object.

ternary_relate(x^t, x^{r1}, x^{r2}, rel) $\rightarrow y$: we propose to extend the formulations above to handle object relationships that involve more than two objects. In this case, x^{r1} and x^{r2} are two reference objects. In ReferIt3D [2], there are two types of ternary relationships: spatial ternary relations (e.g., *between*), and view-dependent relations. Both are resolved with this operation. As an example, the sentence “Facing the

front of the shelf, select the door that is on the right side of it.” yields the program *anchor*(*filter*(*shelf*), ...). Internally, such *anchor* operation will be handled as a ternary relation function: *ternary_relate*(*filter*(*door*), *filter*(*shelf*), *filter*(*shelf*), *anchor-right*). The two reference objects (the reference for the relation “right” and the anchor for “facing”) are the same. Notably, NS3D’s ternary operation can be generalized as a principled solution to any high-arity relations that can be executed based on learned features of the corresponding arity. In our ternary case, it is executed as the following:

$$prob_{i,j,k}^{\text{rel}} = \text{MLP}^{\text{rel}}(f_{i,j,k}^{\text{ternary}})$$

$$y_i = \min\left(x_i^t, \sum_j \sum_k sx(x^{r1})_j \cdot sx(x^{r2})_k \cdot prob_{i,j,k}^{\text{rel}}\right).$$

The NS3D neural program executor composes the above operations and outputs the final object score vector, whose maximum-valued index represents the referred object \mathcal{T} .

3.4. Training

Modules in NS3D can be trained end-to-end with only the groundtruth referred objects as supervision; each can also be trained individually whenever additional labels are available. In this paper, we use a hybrid training objective similar to prior works [2, 20]. Specifically, we use the groundtruth object category to compute a per-object classification loss \mathcal{L}_{oce} (applied to all $prob^c$, where c is the category) and the groundtruth final target object to compute a per-expression loss \mathcal{L}_{ice} . Both loss functions are standard cross-entropy losses. The final total loss, with $\alpha = 1$ in our experiments, is: $\mathcal{L}_{total} = \mathcal{L}_{oce} + \alpha(\mathcal{L}_{ice})$. Practically, we perform a two-stage training: we first pretrain the model with \mathcal{L}_{oce} until convergence, and then train with the full loss.

As a byproduct of NS3D’s modular structure, we gain improved model interpretability. We can validate whether specific programs yield correct outputs at each stage. This can help determine what types of additional data will be valuable. In our experiments, we find that object categorization is the most challenging part of the 3D-REC task.

NS3D does not need the full scene point cloud as its input, and instead only explicitly models a given object set \mathcal{O} . Therefore, we can train on scenes with a small number of objects (10 objects in our experiments) and directly test on scenes with much more objects (88 objects maximum in the test set). This improves training efficiency and reduces the need for annotated 3D objects, which are expensive to acquire in 3D domains. It also enables generalization to more cluttered and complex scenes. In all of our experiments, we train NS3D on scenes with a sparse amount of objects and see no performance drop compared to training with a dense train set. By contrast, baseline methods yield significantly decreased performance under this setting; we present these results in the supplementary material.

*For conciseness, we have used *filter*(*shelf*) as a short-hand notation for *filter*(*scene*(\cdot), *shelf*).

4. Experiments

We evaluate NS3D and compare it to prior work on the ReferIt3D benchmark [2], a 3D referring expression comprehension (3D-REC) task. We specifically focus on the SR3D setting, which tackles spatially-oriented object referential language in 3D scenes. In Section 4.1, we compare performance against baselines and report ablations. In Section 4.2 and Section 4.3, we show experiments on data efficiency and generalization. Finally, in Section 4.4, we present NS3D’s ability to zero-shot transfer to a 3D-QA task.

4.1. 3D referring expression comprehension

In the ReferIt3D task, the input is a set of point clouds, one for each object in the scene, as well as the utterance, and the target object is assumed to be unique. Therefore, models can be evaluated by the accuracy of selecting the correct object. Figure 6 shows an example task instance in ReferIt3D and its NS3D execution trace.

Results. In Table 1, we compare NS3D to baselines. We group methods into two categories: methods that only use the per-object point clouds and methods that model the full scene. The results show that we outperform other object-centric methods and achieve comparable performance with top-performing methods. In addition, we demonstrate state-of-the-art view-dependent accuracy compared to all prior work, with an improvement of 3.6% against the top-performing baseline. NS3D shows close performance between overall and view-dependent accuracy, while all other methods yield a large gap. Note that unlike models such as MVT [18] that explicitly transform and encode point clouds from multiple views to improve multi-view performances, our model simply uses a general high-arity neural network.

Ablations. In Table 3, we first discuss the performance of relational grounding modules. Specifically, since the ReferIt3D dataset does not contain groundtruth labels for object relations, we study the performance of relation grounding modules by evaluating NS3D performance using the groundtruth object classification for all *filter* operations. We see that NS3D achieves almost perfect performance on the task, indicating that it learns relations and high-arity relations well. Our neuro-symbolic approach allows for such diagnostics of model performance: the primary challenge to NS3D is object classification, which can potentially be improved with more object labels.

We next explore the importance of separating object encoders \mathcal{E}^{obj} and \mathcal{E}^{rel} . Having separate object and relation features allows each to specialize in different goals: object classification and relational reasoning. We see that overall performance decreases by 4.6% if they share the same feature encoder. We additionally present results on using an incorrect number of arguments for executing high-arity queries (*i.e.*, spatial ternary relations and view-dependent

	OVERALL	VIEW-DEP.
- WITHOUT 3D SCENE MODELING		
NS3D (OURS)	0.627	0.620
SAT [37]	0.579	0.492
TRANSREFER [16]	0.574	0.499
LANGUAGEREFER [30]	0.560	0.492
3DREFTRANSFORMER [1]	0.470	0.443
REFERIT3D [2]	0.408	0.392
+ WITH 3D SCENE MODELING		
BUTD-DETR [20]	0.670[†]	0.530
MVT [18]	0.645	0.584
3DVG-TRANSFORMER [40]	0.514	0.446
INSTANCEREFER [39]	0.480	0.454
TEXT-GUIDED-GNNS [17]	0.450	0.458

Table 1. NS3D yields the highest overall accuracy on the SR3D task among object-centric methods, and state-of-the-art view-dependent accuracy across all methods.

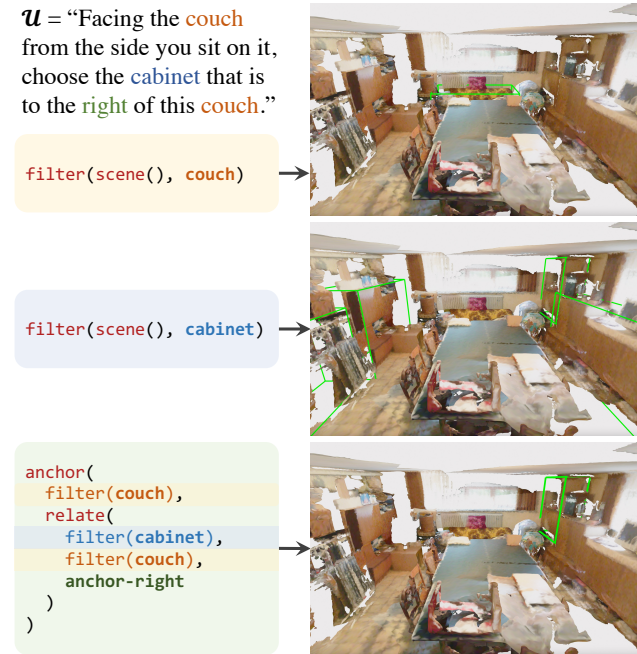


Figure 6. Example of NS3D’s execution trace on a view-dependent 3D-REC task. NS3D returns the correct target cabinet object.

relations) by removing the second reference object. It leads to a 5.8% drop in view-dependent accuracy, which supports the importance of high-arity modules.

4.2. Data efficiency

We report experiments on data efficiency compared to four top-performing prior work on ReferIt3D, two object-centric methods (SAT [37] and TransRefer [16]) and two

[†]We note that BUTD-DETR operates on a more constrained setting, assuming 485 classes instead of the full 607 classes in ReferIt3D.

	0.5%		1.5%		2.5%		5%		10%	
	ALL	V-DEP.	ALL	V-DEP.	ALL	V-DEP.	ALL	V-DEP.	ALL	V-DEP.
NS3D + FULL (OURS)	0.503	0.395	0.576	0.468	0.587	0.505	0.597	0.505	0.612	0.552
NS3D (OURS)	0.426	0.375	0.520	0.424	0.556	0.483	0.591	0.493	0.600	0.527
BUTD-DETR [20]	0.083	0.089	0.158	0.138	0.259	0.223	0.395	0.302	0.528	0.420
MVT [18]	0.161	0.118	0.275	0.199	0.322	0.270	0.380	0.375	0.491	0.426
SAT [37]	0.172	0.149	0.260	0.254	0.298	0.273	0.330	0.309	0.362	0.334
TRANSREFER [16]	0.188	0.152	0.268	0.233	0.305	0.278	0.362	0.380	0.390	0.378

Table 2. Data efficiency results of NS3D compared to prior works, with 0.5%, 1.5%, 2.5%, 5%, and 10% of train data. We report two variations of NS3D, with object classification pretrained on the full dataset and pretrained on the specified data-efficient train set.

	OVERALL	VIEW-DEP.
NS3D W/ GT OBJ. CLS.	0.969	0.823
NS3D W/O SEP. FEAT.	0.581	0.512
NS3D W/O TERNARY ARG.	0.609	0.562
NS3D (FULL)	0.627	0.620

Table 3. Ablation on NS3D with groundtruth object classification results in *filter*, without separation of object and relation features, and without leveraging the correct arity for ternary operations.

methods that model the full 3D scene (BUTD-DETR [20] and MVT [18]). We test on 0.5% (329 examples), 1.5% (987 examples), 2.5% (1,646 examples), 5% (3,292 examples), and 10% of data (6,584 examples) in the train set, with the same full test set. We note that BUTD-DETR [20] uses pretrained object classification results on the full ScanNet dataset [13], while others do not. Hence in Table 2, we report NS3D’s performance on both settings: pretrained object classification on the full ReferIt3D train set (NS3D + Full), and on the smaller train set only (NS3D).

Shown in Figure 7, we see that in both settings, our neuro-symbolic approach significantly outperforms prior work across all data-efficient settings, achieving only small drops in accuracy compared to training on the full train set. NS3D yields 52.0% accuracy with just 1.5% of the train data, with 987 examples only, while all other baselines report accuracy lower than 27.5%. We see this trend persist across data-efficient settings. NS3D sees only a 3.6% gap when using 5% vs 100% of data, while all other methods decrease in performance significantly. NS3D’s accuracy of 59.1% at 5% train data is higher than that of baselines at 100% train data, aside from BUTD-DETR [20] and MVT [18]. This is a significant improvement in the 3D domain, where data annotation is especially labor intensive and expensive.

4.3. Generalization

We present two additional generalization settings and compare our model against prior work in Table 4. The first setting (PAIRS) evaluates performance on unseen object-relation-object pairs. The referring expressions in the train set include the top 5 percent of object-relation-object pairs: *i.e.*, the referred object category, relation type, and the ref-

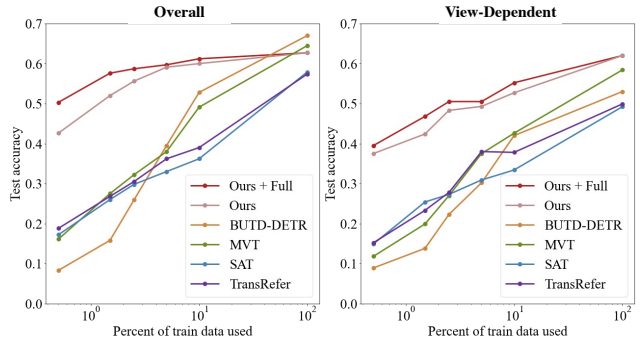


Figure 7. NS3D outperforms prior works by a large margin with 0.5%, 1.5%, 2.5%, 5%, and 10% of train data.

erence object category (*e.g.*, chair-closest-door). The test set contains the bottom 95 percent of object-relation-object pairs in the long-tailed distribution. NS3D does not see a noticeable performance drop, while methods that encode dataset bias by attending over all objects regardless of the functional structures perform poorly in evaluation.

The second setting (SCENE) evaluates model performance on an unseen scene type. The train set includes train examples with all scene types aside from that of “living room”, while the test set only contains examples in living rooms. This is an important generalization setting, as we often want to evaluate models on new environments, without having to additionally label examples with every new scene type. NS3D outperforms all prior work in this setting[‡].

4.4. Zero-shot transfer

Finally, we showcase NS3D’s ability to zero-shot transfer to a new 3D question answering task (3D-QA). Since there are no existing closed-vocabulary 3D-QA datasets in the same domain, we have manually created a small evaluation set of 50 examples for 3D-QA, where the input is a set of objects in the scene, $\mathcal{O} = \{O_1, \dots, O_M\}$, and a question \mathcal{Q} . In contrast to the 3D-REC task, where the output is the target object, the output for 3D-QA is an answer in

[‡]We note that view-dependent accuracy for scene generalization has high variance due to the small amount (36 examples) of test view-dependent examples in living room scenes.

	PAIRS		SCENE	
	ALL	V-DEP.	ALL	V-DEP
NS3D + FULL (OURS)	0.612	0.635	0.563	0.583
NS3D (OURS)	0.599	0.620	0.544	0.611
BUTD-DETR [20]	0.440	0.423	0.515	0.583
MVT [18]	0.420	0.353	0.502	0.500
SAT [37]	0.359	0.380	0.451	0.500
TRANSREFER [16]	0.322	0.344	0.384	0.361

Table 4. Generalization performance to unseen object co-occurrence pairs and scene. NS3D outperforms all baselines.

text form (the vocabulary contains all categories, relations, Yes/No, and integers). The dataset consists of four main types of questions; see Figure 8 for examples of each type. The first are exist-typed questions, which ask whether an object of the specified class and relation exists. The second are count-typed questions, which ask for the number of objects that satisfies the specification. The third are object-typed questions, which ask for object categories, and the last are relation-typed questions, which ask for the relationship between the specified objects. Each type of question has view-dependent and ternary relation variants.

In the semantic parsing stage, NS3D parses the new input questions into symbolic programs by specifying only a handful of prompts for our Codex-based parser (10 sentence-program pairs). By simply specifying one prompt for each type of expected program structure, we gain perfect parsing capabilities of this new program structure. In the 3D feature encoding stage, NS3D can directly re-use learned object and relation grounding modules from 3D-REC for 3D-QA. The 3D object-centric features are the same across both tasks.

The new functional modules introduced in NS3D for 3D-QA task output text answers. The *query_exist* operation is implemented as max over a threshold, and the *query_count* operation as the sum over a threshold, both based on the object score vector. The *query_object* and *query_relation* operations return the category or relation label with the highest prediction probability across labels. Formal definitions for each operation are described in the supplementary material. Note that all modules require no additional training, and are built through composing learned models from 3D-REC.

We show that NS3D can zero-shot transfer across tasks in 3D, with *no additional finetuning or training* of neural networks required. In Table 5, we report accuracy, calculated as the exact match of the text output, for overall performance and for each question type. We compare against NS3D with a finetuned T5 conditional generation model [29] as its semantic parser (NS3D + T5), instead of with Codex (NS3D + Codex). The T5 model is pretrained then finetuned on the same set of examples that Codex received. NS3D with Codex outperforms NS3D with the finetuned T5 model by a large margin, due to T5’s inability to generalize to new words outside of its small train set as well as errors in parsing text into

	ALL	EXIST	COUNT	OBJ	REL
NS3D + CODEX	0.68	0.80	0.67	0.60	0.60
NS3D + T5	0.30	0.40	0.13	0.40	0.30
RANDOM	0.16	0.40	0.07	0.00	0.10

Table 5. NS3D’s zero-shot transfer performance on the 3D-QA task, with comparison of semantic parsers (Codex vs T5), and with a randomly initialized model as baseline.

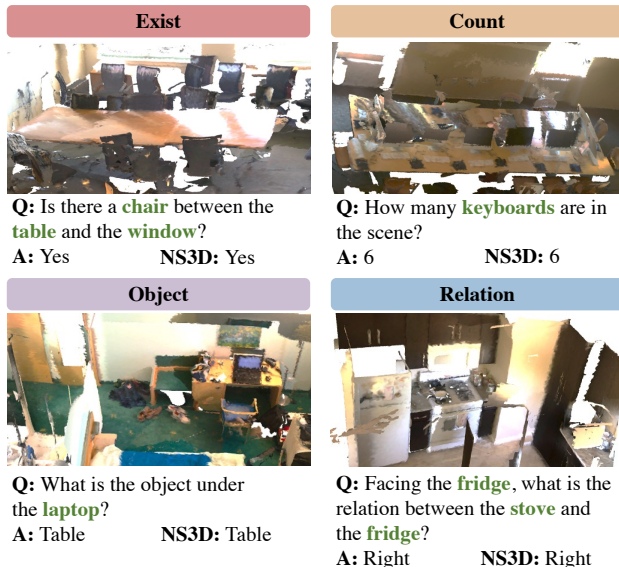


Figure 8. Examples of 4 questions types from the 3D-QA task.

syntactically and semantically correct programs. We also report accuracy from a randomly initialized NS3D model, showing that executing programs with learned modules is indeed significantly more successful in this task. We show more qualitative examples in the supplementary material.

5. Conclusion

We have presented NS3D, a neuro-symbolic model for 3D grounding that leverages compositional programs and modular neural networks to solve complex 3D-REC tasks. It enables strong data-efficiency, generalization to novel data distributions, and zero-shot transfer of 3D knowledge to a new 3D-QA task. We show that we can integrate large language-to-code models with modular neural networks, and accurately parse language into symbolic programs for visual reasoning. We also present a neural program executor that implements high-arity operations effectively to ground complex semantic forms, such as view-dependent anchoring. Together, these components of NS3D form a powerful model for 3D visual understanding. As a future direction, combining NS3D with strong object localization models can potentially enable learning directly from 3D scenes.

Acknowledgments. This work is in part supported by Stanford HAI, NSF RI #2211258, ONR MURI N00014-22-1-2740, AFOSR YIP FA9550-23-1-0127, Amazon, Analog Devices, Bosch, JPMorgan Chase, Meta, and Salesforce.

References

- [1] Ahmed Abdelreheem, Ujjwal Upadhyay, Ivan Skorokhodov, Rawan Al Yahya, Jun Chen, and Mohamed Elhoseiny. 3dref-transformer: Fine-grained object identification in real-world scenes using natural language. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 3941–3950, 2022. [1](#), [2](#), [6](#)
- [2] Panos Achlioptas, Ahmed Abdelreheem, Fei Xia, Mohamed Elhoseiny, and Leonidas Guibas. Referit3d: Neural listeners for fine-grained 3d object identification in real-world scenes. In *European Conference on Computer Vision*, pages 422–440. Springer, 2020. [1](#), [2](#), [3](#), [5](#), [6](#), [13](#)
- [3] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Learning to compose neural networks for question answering. *arXiv preprint arXiv:1601.01705*, 2016. [3](#)
- [4] Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Dan Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48, 2016. [3](#)
- [5] Iro Armeni, Zhi-Yang He, JunYoung Gwak, Amir R Zamir, Martin Fischer, Jitendra Malik, and Silvio Savarese. 3d scene graph: A structure for unified semantics, 3d space, and camera. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 5664–5673, 2019. [3](#)
- [6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. [3](#)
- [7] Daigang Cai, Lichen Zhao, Jing Zhang, Lu Sheng, and Dong Xu. 3djcg: A unified framework for joint dense captioning and visual grounding on 3d point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16464–16473, 2022. [2](#)
- [8] Dave Zhenyu Chen, Angel X Chang, and Matthias Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. In *European Conference on Computer Vision*, pages 202–221. Springer, 2020. [2](#)
- [9] Dave Zhenyu Chen, Qirui Wu, Matthias Nießner, and Angel X. Chang. D3net: A speaker-listener architecture for semi-supervised dense captioning and visual grounding in rgb-d scans, 2021. [2](#)
- [10] Jiaming Chen, Weixin Luo, Xiaolin Wei, Lin Ma, and Wei Zhang. Ham: Hierarchical attention model with high performance for 3d visual grounding. *arXiv preprint arXiv:2210.12513*, 2022. [2](#)
- [11] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. [2](#), [3](#)
- [12] Wenhui Chen, Zhe Gan, Linjie Li, Yu Cheng, William Wang, and Jingjing Liu. Meta module network for compositional visual reasoning. In *WACV*, 2021. [3](#)
- [13] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 5828–5839, 2017. [7](#)
- [14] Mingtao Feng, Zhen Li, Qi Li, Liang Zhang, XiangDong Zhang, Guangming Zhu, Hui Zhang, Yaonan Wang, and Ajmal Mian. Free-form description guided 3d visual graph network for object grounding in point cloud. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3722–3731, 2021. [3](#)
- [15] Chi Han, Jiayuan Mao, Chuang Gan, Josh Tenenbaum, and Jiajun Wu. Visual concept-metaconcept learning. *Advances in Neural Information Processing Systems*, 32, 2019. [3](#)
- [16] Dailan He, Yusheng Zhao, Junyu Luo, Tianrui Hui, Shaofei Huang, Aixi Zhang, and Si Liu. Transrefer3d: Entity-and-relation aware transformer for fine-grained 3d visual grounding. In *Proceedings of the 29th ACM International Conference on Multimedia*, pages 2344–2352, 2021. [1](#), [2](#), [6](#), [7](#), [8](#), [14](#)
- [17] Pin-Hao Huang, Han-Hung Lee, Hwann-Tzong Chen, and Tyng-Luh Liu. Text-guided graph neural networks for referring 3d instance segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1610–1618, 2021. [1](#), [2](#), [6](#)
- [18] Shijia Huang, Yilun Chen, Jiaya Jia, and Liwei Wang. Multi-view transformer for 3d visual grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15524–15533, 2022. [1](#), [2](#), [6](#), [7](#), [8](#), [14](#)
- [19] Drew Hudson and Christopher D Manning. Learning by abstraction: The neural state machine. *Advances in Neural Information Processing Systems*, 32, 2019. [3](#), [4](#)
- [20] Ayush Jain, Nikolaos Gkanatsios, Ishita Mediratta, and Katerina Fragkiadaki. Bottom up top down detection transformers for language grounding in images and point clouds. In *European Conference on Computer Vision*, pages 417–433. Springer, 2022. [1](#), [2](#), [5](#), [6](#), [7](#), [8](#), [14](#)
- [21] Justin Johnson, Bharath Hariharan, Laurens Van Der Maaten, Judy Hoffman, Li Fei-Fei, C Lawrence Zitnick, and Ross Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE international conference on computer vision*, pages 2989–2998, 2017. [3](#)
- [22] Qing Li, Siyuan Huang, Yining Hong, Yixin Chen, Ying Nian Wu, and Song-Chun Zhu. Closed loop neural-symbolic learning via integrating neural perception, grammar parsing, and symbolic reasoning. In *International Conference on Machine Learning*, pages 5884–5894. PMLR, 2020. [3](#)
- [23] Junyu Luo, Jiahui Fu, Xianghao Kong, Chen Gao, Haibing Ren, Hao Shen, Huaxia Xia, and Si Liu. 3d-sps: Single-stage 3d visual grounding via referred point progressive selection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16454–16463, 2022. [2](#)
- [24] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019. [2](#), [3](#), [4](#)
- [25] David Mascharka, Philip Tran, Ryan Soklaski, and Arjun Majumdar. Transparency by design: Closing the gap between performance and interpretability in visual reasoning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4942–4950, 2018. [3](#)

- [26] William Merrill and Ashish Sabharwal. Transformers implement first-order logic with majority quantifiers. *arXiv preprint arXiv:2210.02671*, 2022. 1
- [27] Gabriel Poesia, Oleksandr Polozov, Vu Le, Ashish Tiwari, Gustavo Soares, Christopher Meek, and Sumit Gulwani. Synchromesh: Reliable code generation from pre-trained language models. *arXiv preprint arXiv:2201.11227*, 2022. 3
- [28] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 30, 2017. 4
- [29] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020. 4, 8
- [30] Junha Roh, Karthik Desingh, Ali Farhadi, and Dieter Fox. Language-refer: Spatial-language model for 3d visual grounding. In *Conference on Robot Learning*, pages 1046–1056. PMLR, 2022. 1, 2, 6
- [31] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019. 2
- [32] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008. 2
- [33] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 1, 2
- [34] Johanna Wald, Helisa Dharmo, Nassir Navab, and Federico Tombari. Learning 3d semantic scene graphs from 3d indoor reconstructions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3961–3970, 2020. 3
- [35] Hao Wu, Jiayuan Mao, Yufeng Zhang, Yuning Jiang, Lei Li, Weiwei Sun, and Wei-Ying Ma. Unified visual-semantic embeddings: Bridging vision and language with structured meaning representations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 6609–6618, 2019. 3
- [36] Jiajun Wu, Joshua B Tenenbaum, and Pushmeet Kohli. Neural scene de-rendering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 699–707, 2017. 3
- [37] Zhengyuan Yang, Songyang Zhang, Liwei Wang, and Jiebo Luo. Sat: 2d semantics assisted training for 3d visual grounding. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1856–1866, 2021. 1, 2, 3, 6, 7, 8, 14
- [38] Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B Tenenbaum. Neural-symbolic VQA: Disentangling Reasoning from Vision and Language Understanding. In *NeurIPS*, 2018. 3
- [39] Zhihao Yuan, Xu Yan, Yinghong Liao, Ruimao Zhang, Sheng Wang, Zhen Li, and Shuguang Cui. Instancerefer: Cooperative holistic understanding for visual grounding on point clouds through instance multi-level contextual referring. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1791–1800, 2021. 1, 2, 6
- [40] Lichen Zhao, Daigang Cai, Lu Sheng, and Dong Xu. 3dvg-transformer: Relation modeling for visual grounding on point clouds. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2928–2937, 2021. 1, 2, 6

Supplementary for NS3D: Neuro-Symbolic Grounding of 3D Objects and Relations

The appendix is organized as the following. In Appendix A, we formally define the domain-specific language (DSL) used by NS3D. In Appendix B, we provide dataset details, additional qualitative examples on both 3D-REC and 3D-QA tasks, and an additional experiment on scene complexity generalization, where we train models on scenes with a small number of objects but test on larger and more complex scenes.

A. Domain-Specific Language

In this section, we summarize the value types (Table 6) and function definitions (Table 7) of the domain-specific language used in our paper. The *scene* operation is parameter-free, while other functions take input *object_set*'s and output an *object_set*, represented as the object score vector. Query-type operations take input *object_set*'s and output answers of the target type, such as Boolean values (e.g., "is there a chair?") and concept names (e.g., "what is the type of the object next to the table?"). Existence and counting-related operations involve a score threshold $t = 0.8$, which is a scalar hyperparameter. In our experiment, the threshold t is chosen over a separate 3D-QA dataset based on scenes from the train set, instead of the test set. The *query_object*, *query_relation*, and *query_t_relation* operations are implemented through finding the object or relation label based on object score vectors.

Zero-shot transfer to 3D-QA. NS3D composes learned models on 3D-REC to build new 3D-QA operators in a zero-shot manner, requiring no additional training. The 3D-QA modules can be implemented by reusing the MLPs learned for object and relation classification from the 3D-REC task. Intuitively, let us consider *query_object* in 3D-QA, which takes an object as input and outputs its category. Since we have already learned classifiers for all categories (MLPs used in the *filter* operation), NS3D directly reuses these modules to answer the question: it evaluates all MLP classifiers on the object feature and returns the category with the highest score.

Type	Representation	Semantics
<i>object_set</i>	Object score vectors.	Set of objects selected from a scene.
<i>category</i>	Concept names: <i>table, chair, piano</i> , etc.	Object-level properties.
<i>relation</i>	Concept names: <i>near, left, behind</i> , etc.	(Binary) Relationships between two objects.
<i>t_relation</i>	Concept names: <i>between, anchor-left</i> etc.	(Ternary) Relationships among three objects.
* <i>boolean</i>	Strings: <i>yes, no</i> .	Boolean values.
* <i>integer</i>	Integers: <i>0, 1, 2</i> , etc.	Count of objects.

Table 6. Types in the NS3D domain-specific language. *: Types that are only used in the 3D-QA task.

Signature & Implementation	Semantics
$scene() \rightarrow y: object_set$ Implementation: $y_i = 0$, for all $i \in \{1, 2, \dots, M\}$	Return all objects in the 3D scene.
$filter(x: object_set, c: category) \rightarrow y: object_set$ Implementation: $y_i = \min(x_i, prob_i^c) = \min(x_i, MLP^c(f_i^{obj}))$	Return all objects satisfying a concept c .
$relate(x^t: object_set, x^r: object_set, rel: relation) \rightarrow y: object_set$ Implementation: $prob_{i,j}^{rel} = MLP^{rel}(f_{i,j}^{rel})$ $y_i = \min(x_i^t, \sum_j sx(x^r)_j \cdot prob_{i,j}^{rel})$	Return all objects that satisfy the relationship rel between the object sets.
$ternary_relate(x^t: object_set, x^{r1}: object_set, x^{r2}: object_set, trel: t_relation) \rightarrow y: object_set$ Implementation: $prob_{i,j,k}^{trel} = MLP^{trel}(f_{i,j,k}^{ternary})$ $y_i = \min(x_i^t, \sum_j \sum_k sx(x^{r1})_j \cdot sx(x^{r2})_k \cdot prob_{i,j,k}^{trel})$	Return all objects that satisfy the ternary relationship $trel$ between the object sets.
$anchor(object_set, object_set) \rightarrow object_set$ Implementation: internally handled using $ternary_relate$.	Return all objects that satisfy the relationship anchored on the first object set.
$*query_exist(x: object_set) \rightarrow y: boolean$ Implementation: $y = \begin{cases} yes & \text{if } \max_i(\sigma(x_i)) > t \\ no & \text{if } \max_i(\sigma(x_i)) \leq t \end{cases}$	Return Yes/No corresponding to existence of object in the object set.
$*query_count(x: object_set) \rightarrow y: integer$ Implementation: $y = \sum_i \mathbb{1}[\sigma(x_i) > t]$	Return count of objects in the object set.
$*query_object(x: object_set) \rightarrow c: category$ Implementation: $c = \arg \max_c \left(\sum_i sx(x)_i \cdot MLP^c(f_i^{obj}) \right)$	Return type of object in the object set.
$*query_relation(x^t: object_set, x^r: object_set) \rightarrow rel: relation$ Implementation: $rel = \arg \max_{rel} \left(\sum_i \sum_j sx(x^t)_i \cdot sx(x^r)_j \cdot MLP^{rel}(f_{i,j}^{rel}) \right)$	Return relationship between the object sets.
$*query_t_relation(x^t: object_set, x^{r1}: object_set, x^{r2}: object_set) \rightarrow trel: t_relation$ Implementation: $trel = \arg \max_{trel} \left(\sum_i \sum_j \sum_k sx(x^t)_i \cdot sx(x^{r1})_j \cdot sx(x^{r2})_k \cdot MLP^{trel}(f_{i,j,k}^{ternary}) \right)$	Return ternary relationship between the object sets.

Table 7. Primitive functions defined in the NS3D domain-specific language. *: Functions that are only used in the 3D-QA task. Here, $sx(\cdot)$ is the Softmax function, $\sigma(\cdot)$ is the Sigmoid function, and $\mathbb{1}[\cdot]$ is the indicator function which returns 1 when the expression inside the brackets evaluates to true, and 0 otherwise.

B. Experimental Details and Additional Results

In this section, we first present details for the datasets used in the main text. Then, we provide additional results on the scene complexity generalization task, where we train models on scenes with a small number of objects but test on larger and more complex scenes. Finally, we showcase additional qualitative examples for both the 3D-REC and 3D-QA tasks.

B.1. Dataset

ReferIt3D datasets. For settings where NS3D was trained on the full ReferIt3D dataset, we used the exact SR3D training data for all networks, including 707 scenes with object category annotations and 65,844 query-answer pairs in total.

Data efficiency datasets. We generated data-efficient train sets with randomly sampled 0.5% (329 examples), 1.5% (987 examples), 2.5% (1,646 examples), 5% (3,292 examples), and 10% (6,584 examples) of the train set, with the same full test set from SR3D used for evaluation.

PAIRS and SCENE generalization datasets. We created two new datasets to test generalization ability. Both of the datasets are built on the SR3D train and test set.

The first dataset (PAIRS) evaluates performance on unseen object-relation-object pairs. The referring expressions in the train set include the top 5 percent of object-relation-object pairs: *i.e.*, the referred object category, relation type, and the reference object category (*e.g.*, chair-closest-door). The test set contains the bottom 95 percent of object-relation-object pairs in the long-tailed distribution. The train set and test set consists of 16,200 examples and 10,520 examples respectively.

The second dataset (SCENE) evaluates performance on an unseen scene type. The train set includes train examples with all scene types aside from that of “living room”, while the test set only contains examples in living rooms. The train set and test set consists of 57,125 examples and 1,320 examples respectively.

3D-QA dataset. We manually created a small evaluation set of 50 examples for the 3D-QA task, based on the test set of ReferIt3D [2]. The input is a set of objects in the scene, $\mathcal{O} = \{O_1, \dots, O_M\}$, and a question \mathcal{Q} . In contrast to the 3D-REC task, where the output is the target object, the output for 3D-QA is an answer in text form (the vocabulary contains all categories, relations, Yes/No, and integers). The dataset consists of four main types of questions created from the following templates:

Existence-typed questions:

- Is there a [Object] [Relation] [Object]? A: Yes/No
- Is there a [Object] [Relation] [Object] and [Object]? A: Yes/No
- Facing [Object], is there a [Object] [Relation] [Object]? A: Yes/No

Counting-typed questions:

- How many [Object] are in the scene? A: Integer
- How many [Object] are [Relation] [Object]? A: Integer

Object-typed questions:

- What is the item [Relation] [Object]? A: [Object]
- What is the item [Relation] [Object] and [Object]? A: [Object]
- Facing [Object], what is the item [Relation] [Object]? A: [Object]

Relation-typed questions:

- What is the relationship between [Object] and [Object]? A: [Relation]
- Facing [Object], what is the relationship between [Object] and [Object]? A: [Relation]

B.2. Scene Complexity Generalization

For all experiment results reported in the main text, NS3D was trained on examples with only 10 objects given in the scene, and evaluated on the full test set with up to 88 objects in the scene. NS3D is able to show this scene complexity generalization, as it does not need the full scene point cloud as its input and instead only explicitly models a given object set and relations between specified objects. This improves training efficiency, reduces the need for annotated 3D objects, which are expensive to acquire in 3D domains, and enables generalization to more cluttered scenes.

We show that baselines methods cannot generalize as NS3D does, and yields significantly decreased performance when trained on 10 objects per scene and evaluated on more complex scenes. In Table 8, we see that NS3D outperforms prior works by a large margin in this setting. We did not test BUTD-DETR, because BUTD-DETR explicitly encodes the full 3D scene as input, with all objects given in train and test, and hence does not directly apply to this partial scene setup.

B.3. NR3D Results

We report results on NR3D, the natural language variant of ReferIt3D. While NS3D does work on natural language, as Codex can parse NR3D input into programs, Codex parsing yields 91 distinct function modules and 5892 concepts, resulting in a separate long-tailed problem. Hence, we ran additional experiments on a subset of NR3D, by restricting utterances to those that parse to the same set of functions and concepts in SR3D, which yields 3659 train examples and 1041 test examples.

We train NS3D as well as top-performing baselines, and see that NS3D significantly outperforms prior work (Table 9). This suggests that NS3D can learn from natural language data in a data-efficient way. Examples from this NR3D subset include noisy natural language such as “The picture above the bed with the laptop on it.” and “The monitor that you would class as in the middle of the other two”; both exhibit noisy natural language, with more complex underlying programs than the SR3D training set.

B.4. Qualitative Examples

In Figure 9, we show additional examples of the ReferIt3D 3D-REC task in SR3D, with examples of binary and ternary relations. In Figure 10, we present comparisons of NS3D against baselines on view-dependent examples, with the green outline indicating correct selection and red outline indicating incorrect selection. We see that NS3D is able to outperform prior work in disambiguating the target referred object.

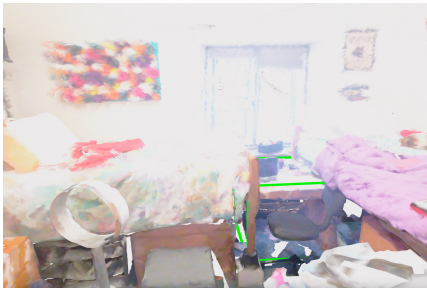
In Figure 11, we present additional qualitative examples of NS3D on the 3D-QA task. We see examples of success cases in green and failure cases in red for NS3D in the zero-shot transfer setting.

	OVERALL	VIEW-DEP.
NS3D (OURS)	0.627	0.620
MVT [18]	0.405	0.396
SAT [37]	0.444	0.415
TRANSREFER [16]	0.360	0.344

Table 8. Generalization results from sparse scenes to dense scenes.

	OVERALL	VIEW-DEP.
NS3D (OURS)	0.526	0.432
BUTD-DETR [20]	0.382	0.331
MVT [18]	0.430	0.324
SAT [37]	0.329	0.248
TRANSREFER [16]	0.360	0.286

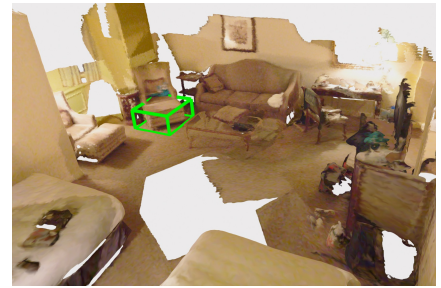
Table 9. Results on a constrained version of NR3D from the ReferIt3D datasets.



Instruction: Select the **desk** that is supporting the laptop.



Instruction: Looking at the front of the bed, select the **nightstand** that is on the right side of it.



Instruction: Select the **ottoman** that is in the center of the desk and the end table.



Instruction: Select the **microwave** that is close to the kitchen cabinets.



Instruction: Facing the front of the couch, choose the **towel** that is on the left side of it.



Instruction: Choose the **suitcase** that is in the middle of the bathtub and the cabinet.

Figure 9. Additional examples of the input language instruction, scene, and the target output in the ReferIt3D 3D-REC task.

Instruction: Facing the front of the couch, choose the **table** that is on the left of it.

Instruction: Looking at the front of the refrigerator, choose the **cabinet** on the right of it.

Instruction: Facing the front of the file cabinet, select the **chair** that is to the right of it.

NS3D



BUTD-DETR



MVT



SAT



Trans Refer



Figure 10. Comparison between NS3D and baselines on the 3D-REC task, with the green outline indicating correct selection and red outline indicating incorrect selection.

Existence-typed questions



Q: Is there a **laptop** next to the **backpack**?

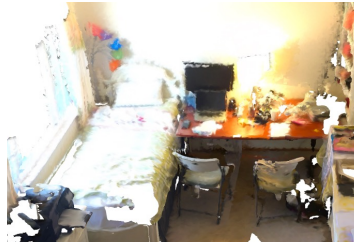
A: No **NS3D:** No



Q: Facing the **toilet**, is there a **sink** to the left of the **toilet**?

A: Yes **NS3D:** Yes

Counting-typed questions



Q: How many **chairs** are next to the **lamp**?

A: 2 **NS3D:** 2



Q: How many **kitchen cabinets** are in the scene?

A: 5 **NS3D:** 5

Object-typed questions



Q: What is the item between the **door** and the **couch**?

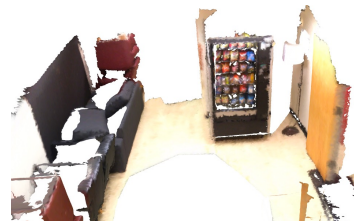
A: Shelf **NS3D:** Table



Q: What is the item besides the **dining table**?

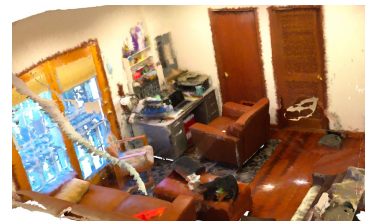
A: Chair **NS3D:** Chair

Relation-typed questions



Q: What is the relationship between the **vending machine** and the **lamp**?

A: Near **NS3D:** Near



Q: Facing the **door**, what is the relationship between the **table** and the **door**?

A: Left **NS3D:** Right

Figure 11. Qualitative examples of NS3D on the 3D-QA task. Examples of success cases are marked in green, while failure cases are marked in red.