ReferIt3D: Neural Listeners for Fine-Grained 3D Object Identification in Real-World Scenes Supplemental Material

Panos Achlioptas[1], Ahmed Abdelreheem[2], Fei Xia[1],
Mohamed Elhoseiny[1,2], Leonidas Guibas[1]

[1] Stanford University
[2] King Abdullah University of Science and Technology
[1]{panos, feixia, elhoseiny, guibas}@cs.stanford.edu,
[2]{ahmed.abdelreheem, mohamed.elhoseiny}@kaust.edu.sa

Contents of Supplemental

1 Building Nr3D Details

1.1 Making Stimuli

Nr3D is comprised by a total of 41503 utterances describing objects belonging in one 76 fine-grained object classes in 5878 communication contexts (unique sets {S, I}, where S denotes a specific scene, and I the (single) fine-grained class of the contrasted objects of S). These communication contexts were created by considering all 707 scenes of ScanNet [2] with all their fine-grained annotated objects classes. Concretely, a context {S, I} of Nr3D satisfies:

1. $2 \leq |o \in S \cap \text{class-of(o)} == I| \leq 6$. In words, the contrasted objects are more than 1 but not more than 6, and they are of the same fine-grained object class.
2. There exist 5 or more scenes, S, for any given I, for which the above condition is satisfied.
3. I is not a structural object class ('wall', 'floor', 'ceiling'), nor a part of an object ('doorframe', 'stair rail', 'closet wall'), nor an object class that tends to have vague or poorly annotated object instances ('object', 'decoration', 'clothes', 'clothing').

1.2 Representing ScanNet Scenes on AMT

To create Nr3D, we utilized the web interface presented in Fig 1. The 3D scene shown to users (acting as speakers or listeners) was a decimated mesh representation of a ScanNet scene. The high resolution mesh and low resolution (decimated) mesh are obtained from ScanNet[2] dataset. The decimated mesh is then UV-unwrapped to create texture mapping. Texture is mapped from high resolution ScanNet mesh to decimated mesh mentioned above and packed into GLTF2 format. This way, we can load the decimated mesh in browser fast while keeping high visual quality of the mesh. The users can navigate the scene via rotation, pan and zoom in any place of the given scene. The rendering is done in real-time through a web browser.

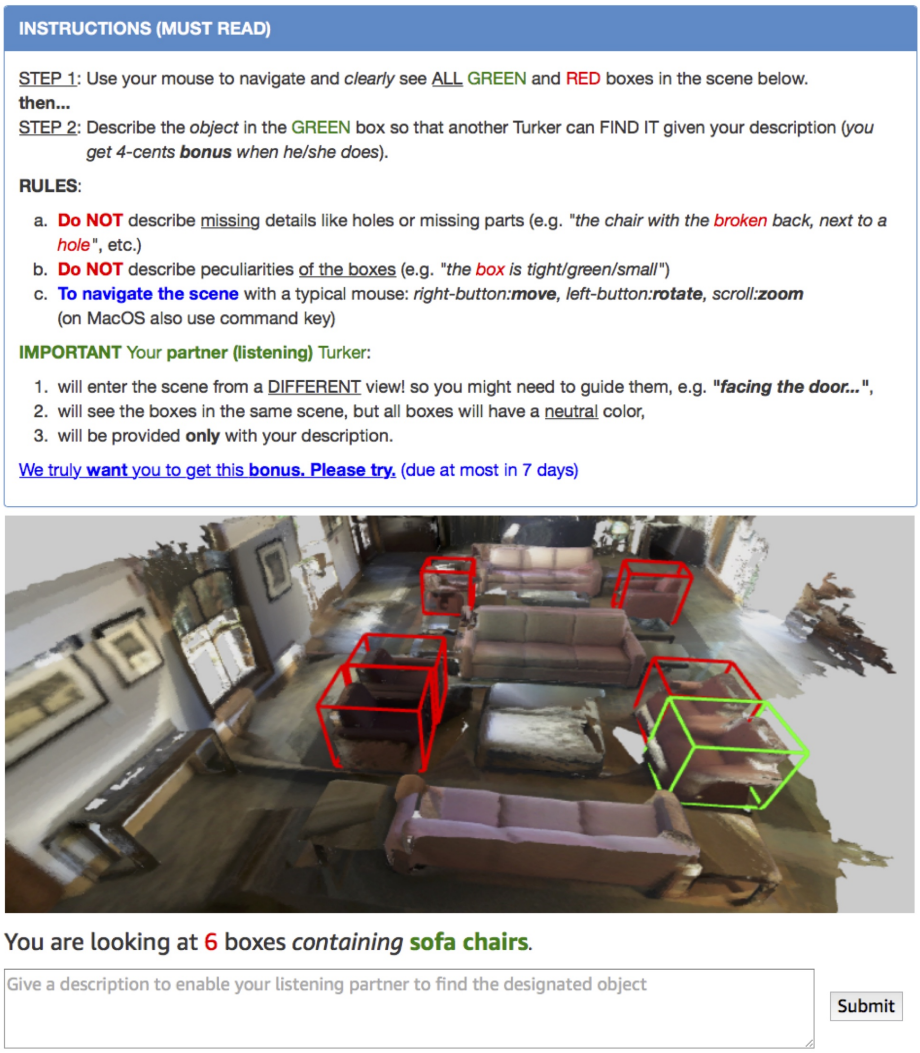

Fig. 1. Snapshot of the interface we used in Amazon Mechanical Turk to collect human utterances while building the **Nr3D**. The Turkers were instructed to pay attention on *all* objects in highlighted boxes, while ignoring any sampling artifacts (e.g., holes or broken pieces of the objects). Furthermore, we *motivated the Turkers* to be effective by providing them a financial bonus *(*50% *of their base-pay)* each time their produced utterance enabled the paired listener to guess the target.

2 Spatial Reference in 3D

Sr3D is built on top of ScanNet [2]. In this section we will discuss the generation method for each spatial relation and how we created human-like utterances out of these relations. In Table 1, we provide the number of unique communication contexts for each relation type.

Spatial Relation	|Contexts|
closest	17126
farthest	16875
between	3569
front	703
behind	113
left	518
right	546
supporting	390
supported by	357
above	960
under	629
Total	**41786**

Table 1. Detailed statistics of Sr3D. For each spatial relation we report the total number of unique tuples (communication contexts) it creates in ScanNet.

2.1 Horizontal Proximity Based Relations

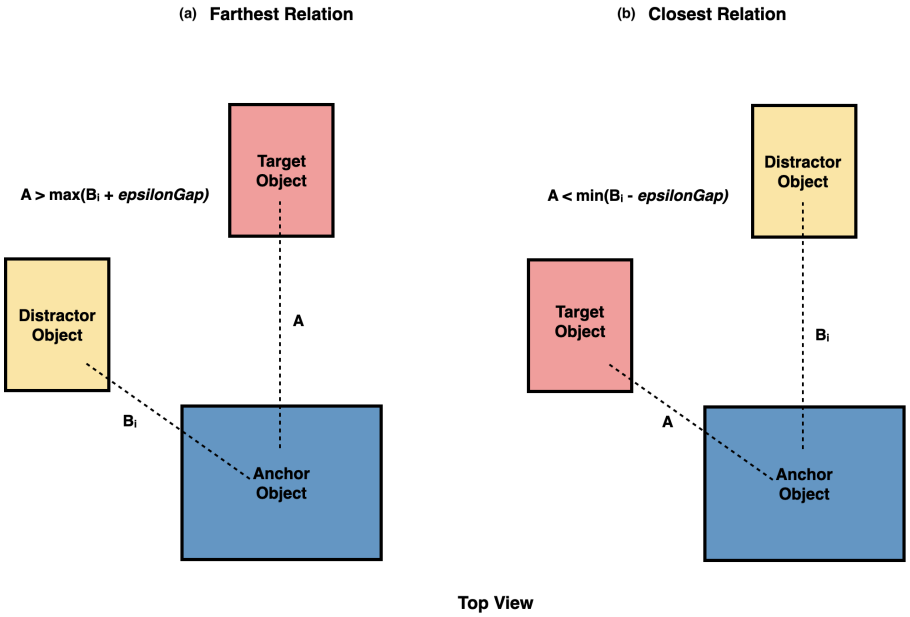

Fig. 2. This figure shows the horizontal (farthest/closest) relations. In (a), the target is the farthest from the anchor object. In (b), it is the opposite. The farthest target to the anchor object should be at a distance greater than *epsilonGap* from the distance between the farthest distractor object and the anchor object.

This type of relations describes what the nearest (closest) and the farthest target objects are according to a certain unique anchor object in the scene. To generate such relations, we get the list of all anchor objects. Then for each anchor object and each target fine-grained class, we calculate the pairwise distance between each same-class target object and this anchor. If the farthest target object to this anchor object is at a distance of *epsilonGap* greater than of the second farthest target object, this combination of the anchor and the farthest target object is used as a relation. Similar logic is used to get the closest relations (See Fig. 2).

2.2 Support Relations

The support relations describe whether a target object is supporting (holding) or supported by (held by) an anchor object. To be able to generate these relations for an anchor object, we need to first get the target objects that lie in the vicinity

of that anchor top/bottom surface. Then, we find the target objects that their top/bottom surfaces touch the anchor's bottom/top surface respectively (See Fig.3). To check if an object is in the vicinity of the anchor object or not, we first look at the two objects' 2D bounding boxes in the top view then we calculate the intersection area between them and the ratio of the object's area to the intersection area should be greater than a certain threshold. For an object to be touching an anchor object, the difference in their bottom/top or top/bottom z surfaces is within a small range.

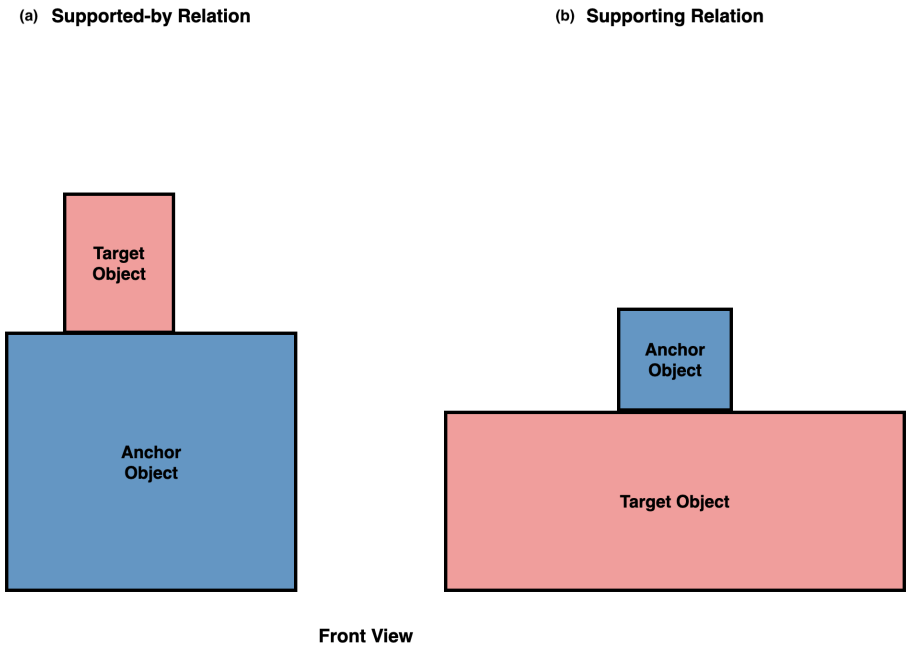

Fig. 3. Example of support relations. In (a), the target is supported by the anchor object and in (b) the target is supporting/holding the anchor object.

2.3 Vertical Proximity Based Relations

These relations represent whether a target object is considered above or below the anchor object without touching each other. The generation method for this type of relations is quite similar to the support relations but we make sure that the target and the anchor objects do not touch each other (See Fig.4.)

2.4 Between Relations

The between relations describe the target objects that lie between two anchor objects (see Fig. 5). To generate such types, we consider all possible pairs of

Fig. 4. Examples of vertical (above/below) relations. In (a), the target is above the anchor object and in (b) it is the opposite. The target and the anchor objects should not be touching each other.

anchor objects. For each anchor pair, we look at the anchors' 2D bounding boxes in the top view (XY axes) and we find the convex hull of those 2D bounding boxes. We search for target objects that satisfy the following conditions: (a) they do not intersect with the two anchors; (b) they exist solely inside the convex hull where none of their distractors are found inside; and (c) intersect with each of the two anchors in the z-axis coordinates.

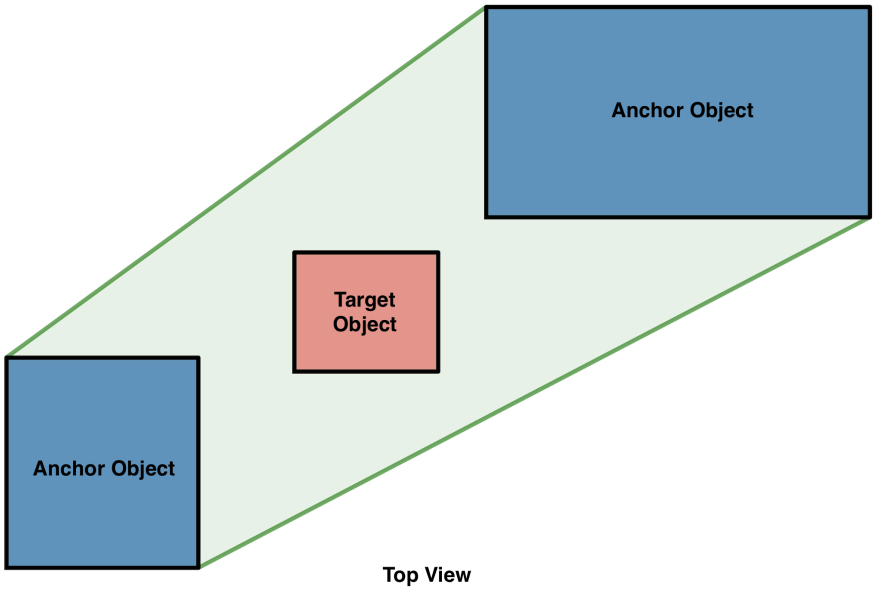

Fig. 5. Example of a between relation. The green shaded area is the area where a target object should be found to be considered as being between the two anchors. This shaded area is the convex hull of the two anchor bounding boxes in the top view. None of the target object's distractors should be inside the shaded area.

2.5 Allocentric Relations

These relations indicate where a target object might exist with respect to the anchor orientation. For example, the armchair (target object) that is at the right of the TV (anchor object). For generating allocentric relations, we need to know: (a) whether the anchor objects have an intrinsic front view (e.g., armchair) or not (e.g., stool); and (b) the orientation of the objects in ScanNet. For (a), we used the annotations of PartNet to extract if a chair has a back or not, so as to define which ShapeNet chair models we will use. Also, we manually annotated several ShapeNet models covering 33 categories in total. For (b) we utilized the Scan2CAD [1] annotations that provide 9DOF alignments between ShapeNet models and ScanNet objects. For every anchor object that has an intrinsic front view, we create four oriented sections (regions) (see Fig. 6) and we try to find the objects that solely occupy an oriented section where no distractors of the same object class co-exist in there. For an object to occupy solely an anchor's oriented section, the ratio of its points inside the section over its total number of points should be greater than a certain threshold (occupancy threshold).

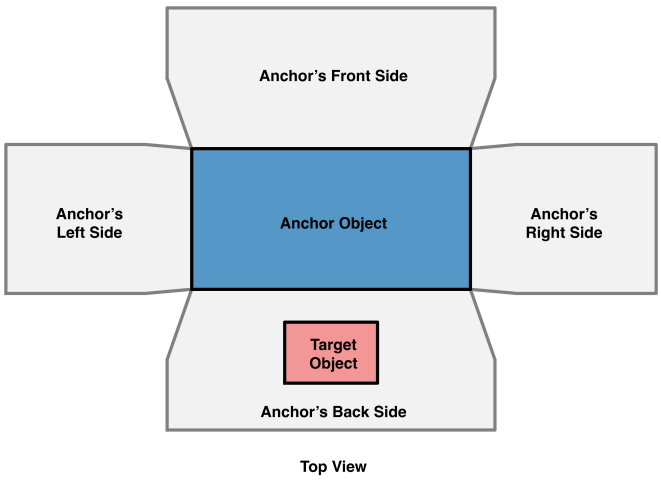

Fig. 6. Allocentric relations generation. This figure shows how we determine where the target object might exist with respect to one of the four oriented sections of the anchor (front, back, left, and right). In this example the target object is at the back of the anchor object.

2.6 Creating Human-like Utterances

To create the human-like utterances out of the spatial references, we have created template sentences for each spatial relation type. We chose to sample at most 2 utterances for each spatial relation by replacing the "target" and "anchor" placeholders in the template sentences with the relation's target and the anchor instance types respectively.

3 Implementation Details

We use 4 graph-convolutional layers for DGCN each producing an intermediate representation of 128 dimensions. D_L, D_V are also 128-dimensional each. We set the α and β hyper-parameters controlling the contribution of the object and text classification losses in the total loss to 0.5 each. To process the linguistic information in our networks, we use a uni-directional LSTM cell [3] with 128 hidden units and word embeddings of $64D$ that were randomly initialized from unit-normal Gaussian. We note that initializing the embeddings with a $100D$ GloVe [5] embedding pre-trained on the 6B Wikipedia 2014 corpus did not give any significant performance boost. For the object referential loss, we use an MLP([128, 64, 1]) network. We sample 1024 points from the point cloud of each segmented object before passing it to the object encoder.

3.1 Preprocessing utterances

We preprocess the collected human utterances by i) lowercasing, ii) tokenizing by splitting off punctuation, iii) replacing tokens that appear less than three times in the training split with a special symbol marking an unknown token (UNK). Furthermore, we ignore all utterances comprised by more than 24 tokens (99th percentile) and those for which the human listener in the underlying trial did not guess correctly the target.

3.2 Training details

We use ADAM [4] with an initial learning-rate of 0.0005 and $\beta_1 = 0.9$ across all our experiments. We train each model for a *maximum* of 100 epochs. We use the test-set to evaluate performance at the end of each training epoch and stop the training if we encounter 10 consecutive training epochs without improvement in terms of test-accuracy. Our batch-size is 32 for all experiments, except for when we train a model with Nr3D and Sr3D utterances (simultaneously) where we use 64 examples in each batch. Last, we use a learning-rate scheduler that reduces the learning rate by a multiplicative factor of 0.65 every 5 consecutive epochs that the test-accuracy did not improve.

4 Listening Ablations

We show the effect of using different values of k nearest neighbor-graph nodes in the DGCN. We trained **ReferIt3DNet-V2** with the default architecture on Nr3D dataset and in Figure 7, we report the listening accuracies.

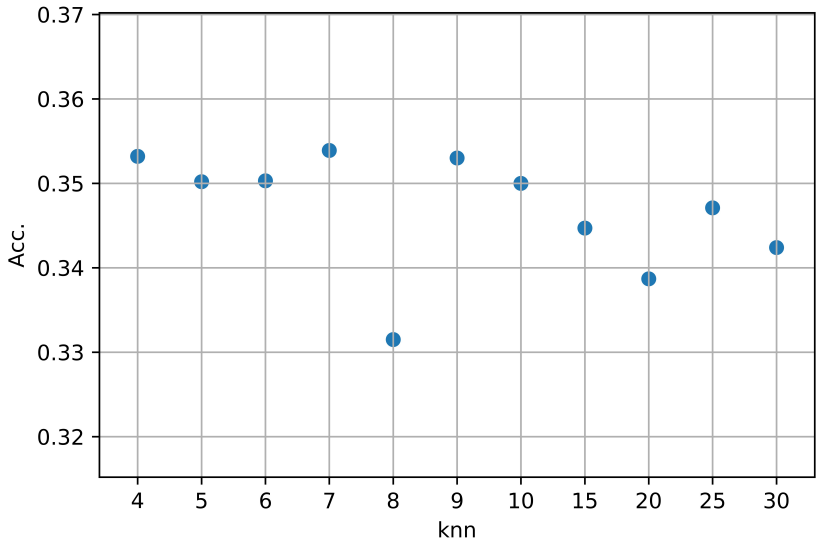

Fig. 7. Reporting the Nr3D test accuracy upon training the default **ReferIt3DNet-V2** using different values of k nearest neighbor-graph nodes in the DGCN.

References

1. Avetisyan, A., Dahnert, M., Dai, A., Savva, M., Chang, A.X., Nießner, M.: Scan2cad: Learning cad model alignment in rgb-d scans. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2019)
2. Dai, A., Chang, A.X., Savva, M., Halber, M., Funkhouser, T., Nießner: Scannet: Richly-annotated 3d reconstructions of indoor scenes (2017)
3. Hochreiter, S., JüRgen, S.: Long short-term memory. In: Neural Computation (1997)
4. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. CoRR **abs/1412.6980** (2014)
5. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Empirical Methods in Natural Language Processing (EMNLP) (2014)