Multi-View Transformer for 3D Visual Grounding

Shijia Huang Yilun Chen Jiaya Jia Liwei Wang
The Chinese University of Hong Kong
{sjhuang,ylchen,leojia,lwwang}@cse.cuhk.edu.hk

Abstract

The 3D visual grounding task aims to ground a natural language description to the targeted object in a 3D scene, which is usually represented in 3D point clouds. Previous works studied visual grounding under specific views. The vision-language correspondence learned by this way can easily fail once the view changes. In this paper, we propose a Multi-View Transformer (MVT) for 3D visual grounding. We project the 3D scene to a multi-view space, in which the position information of the 3D scene under different views are modeled simultaneously and aggregated together. The multi-view space enables the network to learn a more robust multi-modal representation for 3D visual grounding and eliminates the dependence on specific views. Extensive experiments show that our approach significantly outperforms all state-of-the-art methods. Specifically, on Nr3D and Sr3D datasets, our method outperforms the best competitor by 11.2% and 7.1% and even surpasses recent work with extra 2D assistance by 5.9% and 6.6%. Our code is available at https://github.com/sega-hsj/MVT-3DVG.

1. Introduction

Visual Grounding (VG) aims to ground a natural language description to the target object. The field has made tremendous progress in 2D images [18, 22, 30, 35]. With the rapid development of 3D sensors and 3D vision technology, 3D visual grounding has recently attracted more attention, with wide applications including vision-language navigation [31, 39], intelligent agents [26, 32], and autonomous vehicles [11, 21]. Compared to 2D visual grounding, the 3D task has more complex input data (e.g., sparse point clouds) and more variant spatial relationships, switching the output from grounding to 2D regions to 3D objects.

Recent works [1, 3, 14, 25, 33, 36] mainly follow a two-stage scheme, i.e., first generating all candidate objects in 3D scene and then selecting the most matched one. These two-stage approaches in 3D visual grounding are tailored from 2D visual grounding methods [20, 34], which have not considered the unique property of 3D data. Therefore, in this paper, we bring the community's attention back to studying the intrinsic property of 3D data to break through the 3D visual grounding research.

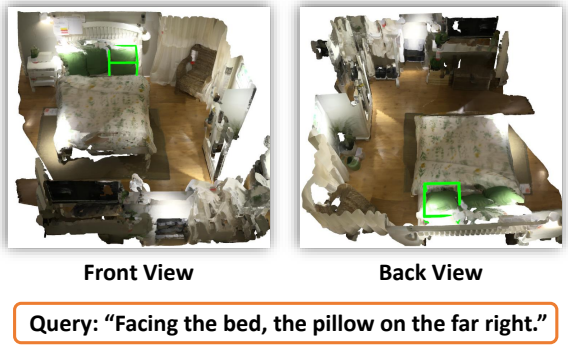

Figure 1. An example from Nr3D [1]: when the view is changed, the position information in 3D scene can be quite different. The mentioned "pillow" exhibits changed positions, i.e., 3D coordinates, when the view is changed from the front to the back.

Position information is essential to locate an object. The visual grounding aligns visual position information in the scene (e.g., coordinates of objects) with the location described in natural language (e.g., "left" or "right"). Unlike 2D images with static views, 3D scenes can freely rotate to different views. These dynamic view changes make 3D visual grounding more challenging. The view changes will directly affect the position encoding, e.g., coordinates of objects, thus bringing more difficulty in representing 3D objects. For example, as shown in Fig. 1, the rightmost pillow in the front view becomes the leftmost one in the back view. This discrepancy commonly exists in real applications: when a robot or an intelligent agent is navigating in the 3D scene, its view is usually different from the speaker's or commander's view. Such view discrepancy can bring more ambiguity in identifying the location of targets. To alleviate this view issue, when Referit3D [1] were collecting data, they split the data into view-dependent and view-independent categories according to whether the description was constrained to the speaker's view. They

have provided additional view guidance, e.g., "Facing the bed" in the example of Fig. 1, to those view-dependent data. In practice, the view constraint is usually implicit, and we cannot always rely on speakers to provide augmented descriptions to solve the problem. Especially, the robot or the intelligent agent should be able to ground to the target object no matter how its view is different from the speaker's. Thus, developing a robust approach to view changes becomes imperative in 3D visual grounding.

Existing methods [1, 10, 14, 25, 33, 36, 38] have paid little attention to the issue as mentioned above and mainly learned the grounding task under specific views. The vision-language correspondence learned in this way can easily fail once the view changes. An intuitive solution to alleviating the problem is to add augmented rotation data under different views during training. However, as we tried and tested (see Tab. 4 in our experimental section), such a straightforward method does not bring the improvement. This result can be understood, i.e., although novel views of the 3D scene can be augmented in the training stage, visual features under specific views are not accurately aligned with the position information embedded in the language description. An alternative is to ask the model to predict the view attached to the language query, say, the speaker's view, and then align the query with the visual features under the predicted view. However, this will incur a lot more costs for view prediction, and if the model predicts the wrong view, the alignment between the two modalities will inevitably be damaged. Experiments provided by LanguageRefer [25] confirmed this point, i.e., even if they manually correct the view of all training data to the speaker's view, the performance of the model on unseen test data with view discrepancy still cannot be improved. Therefore, in this paper, we turn to the exploration of learning better multi-modal representations for 3D visual grounding, which are robust to view changes, instead of augmenting view data or predicting views.

In this paper, we propose a Multi-View Transformer (MVT) for 3D visual grounding to learn such a view-robust representation. Given the 3D scene under a specific view, our MVT projects it to a multi-view space by rotating the original view with equal angles. Our approach simultaneously models the position information within the 3D scene under different views and aggregates all information to learn a view-robust representation by eliminating the dependence on the starting view. Furthermore, to reduce the computational cost of modeling multi-view space, we decouple the 3D object representation computation into computing point clouds features and object coordinates separately to share the point clouds features across views. Besides, with the rich and fine-grained category labels provided by Nr3D [1] and Sr3D [1], we make full use of these language priors and propose a language-guided object classification task to enhance the object encoder further. Extensive experiments have been conducted on three mainstream benchmark datasets Nr3D [1], Sr3D [1], and ScanRefer [3]. Our MVT outperforms all state-of-the-art methods by a large margin. Specifically, on Nr3D and Sr3D, MVT not only outperforms the best competitor by 11.2% and 7.1% respectively but also surpasses the method [33] with extra 2D assistance by 5.9% and 6.6%.

2. Related Work

3D Visual Grounding. The task of 3D visual grounding aims to ground objects described by natural language in 3D scenes. Referit3D [1] and ScanRefer [3] have built benchmarks and baselines for the 3D visual grounding task with language query annotations on ScanNet [7] dataset. Recent works [1, 3, 14, 25, 33, 36] solve the 3D grounding tasks following a two-stage scheme. In the first stage, 3D object proposals are generated either with ground truth [1] or by a 3D object detector [3, 16]. In the second stage, the task is modeled as a matching problem where 3D visual features are fused with the text features of the query language description to predict the matching scores. Referit3D [1] uses GRU [6] to extract text features, and makes use of GNN [27] to model the relationship between objects. With the widespread of transformers [29], recent works [14, 25, 33] have explored transformers for feature extraction and feature fusion. Among them, InstanceRefer [36] conducts multi-level contextual referring by considering attributes and local relations. LanguageRefer [25] replaces point cloud features with predicted labels of objects, transforming the multi-modal task into a language modeling problem. SAT [33] has conducted joint training of 2d grounding and 3d grounding, improving the performance significantly by cross-modal knowledge transfer. 3DVG-Transformer [38] proposes a coordinate-guided contextual aggregation module and a multiplex attention module to fully utilize the contextual clues in the 3D scene. FFL-3DOG [10] uses language and visual scene graphs to guide the feature matching between language and point cloud. TransRefer3D [14] designs an entity-aware attention module and a relation-aware attention module to conduct fine-grained cross-modal feature matching. Different from previous works, we start by studying the intrinsic problem of 3D visual data and build a view-robust multi-modal representation to solve the problem.

Learning with Natural Language Supervision. In parallel, there are also a series of works [8, 13, 17, 24, 37] to learn the visual model guided by natural language features, with the target of learning transferrable models from language. Among them, VirTex [8] pre-trains the network using dense captions (e.g., COCO Captions [4]) to learn visual representations and transfer them to downstream visual recognition tasks. CLIP [24] performs the natural language

supervision on a sufficiently large dataset collected from the Internet and demonstrates its powerful zero-shot transfer capability. In this paper, we leverage fine-grained object labels to enhance object encoders to improve the alignment between object features and language queries.

Multi-View Learning. In 3D computer vision, recent works have focused on learning better representation by rendering objects from different views. MV3D [5] encodes the sparse 3D point cloud with a compact multi-view (e.g., bird's eye view and front view) representation for the 3D detection task [12]. MVCNN [28] completes the 3d object classification by rendering a large number of 2d pictures from 3d objects. These works project 3D point clouds onto different 2D planes to enhance visual features. Unlike them, we project the positional information of the 3D scene into a multi-view space, thereby learning a view-robust representation for better vision-language alignment.

3. Method

In this section, we introduce the proposed Multi-View Transformer (MVT) for 3D visual grounding. Fig. 3 shows the whole framework, which contains object encoding, language encoding, multi-modal feature fusion, and multi-view aggregation modules. In this section, we first illustrate our multi-view 3D visual grounding framework and then provide details of each component.

3.1. Multi-View 3D Visual Grounding

The key idea of our model is to learn a multi-modal representation independent from its specific starting view. Given the 3D scene S under an arbitrary view and a natural language query Q, we extend S to N different views $\{\mathrm{S}^1, \ldots, \mathrm{S}^N\}$ through rotations, given as:

$$\mathrm{S}^j = R(\theta_v^j) \times \mathrm{S}, \tag{1}$$

$$R(\theta) = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix}^T, \tag{2}$$

where $R(\theta)$ is the rotation matrix, and T is the transpose. $R(\theta) \times \mathrm{S}$ means rotating the 3D scene S clockwise by θ degrees along the Z-axis. θ_v^j is the view-rotation angle of the j-th view. We adopt the equal angle rotation, which is:

$$\theta_v^j = \frac{2\pi(j-1)}{N}, j \in \{1, \ldots, N\}, \tag{3}$$

where N is the number of total views, usually set to be $\{1, 2, 4, 8\}$. Fig. 2 shows θ_v with different view settings. θ_v^1 equals to 0 hence S^1 is the same as S. Therefore, the original 3D visual grounding task VG under certain view S is now extended to learn a more robust multi-view representation, which is modeled as:

$$\mathrm{VG}(\mathrm{S}, Q) = \mathrm{g}(f(\mathrm{S}^1, Q), \ldots, f(\mathrm{S}^N, Q)), \tag{4}$$

where $f(\cdot)$ is a shared network for feature extractions under different views. $f(\cdot)$ takes into account the position information of different views in the multi-view space. $\mathrm{g}(\cdot)$ is an order-independent function for information aggregation. Therefore, Eq.(4) has the intriguing property, i.e., when we change the starting view from S to any other in $\{\mathrm{S}^1, \ldots, \mathrm{S}^N\}$, the final multi-view representation of $\mathrm{VG}(\mathrm{S}, \mathrm{Q})$ keeps invariant. This property meets our goal of building a robust representation that is independent of its initial view by projecting S to the multi-view space of $\{\mathrm{S}^1, \ldots, \mathrm{S}^N\}$.

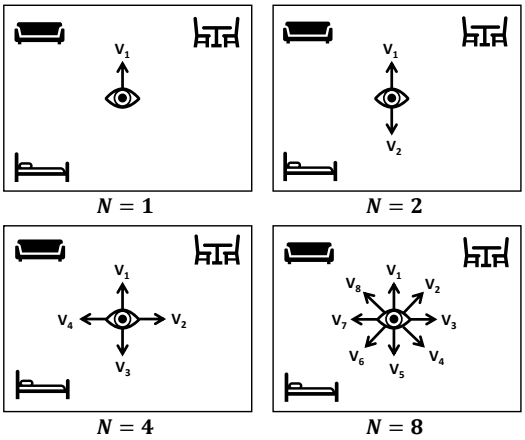

Figure 2. The illustration of multi-view generation. N is the view number, V_1 is the initial view of scene S. We generate multiple views by the equal angle rotation, which divides the 2π view space equally into N parts.

3.2. Object Feature Encoding

Following previous settings [1, 3], we assume to have access to M candidate objects in the scene S under a specific view. Objects are either generated from the ground truth as in Referit3D [1] or predicted by a 3D object detector as in ScanRefer [3]. 3D object encoding extracts features from 3D point cloud data, which is the main computing bottleneck, and it is more costly to encode features for multiple views. Therefore, we decoupled the 3D object feature extraction process into two steps, i.e., computing *point clouds features* and *multi-view positional encoding*. This decoupling can significantly save the computational cost since different views can share the same point cloud features outputted from the first step. Finally, the object features can be calculated by simply combining both.

Point Cloud Features. We sample 1024 points of point clouds for each object i in the form of RGB-XYZ, denoted as $pc_i \in \mathbb{R}^{1024\times 6}$. Then point cloud features $\{x_1, \ldots, x_M\}$

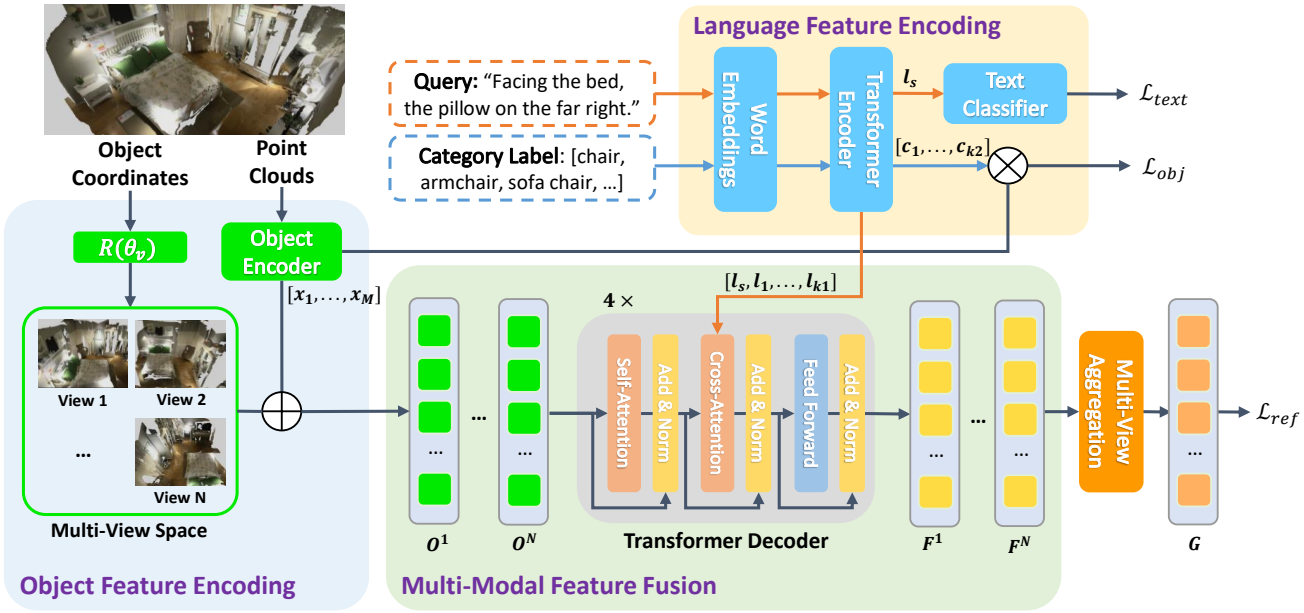

Figure 3. The framework of the proposed multi-view transformer (MVT) for 3D visual grounding. We project the 3D scene to a multi-view space by the equal angle rotation. For each view, object features and language features are fused by the transformer decoder [29]. We finally aggregate information from multi-view space to form a view-robust representation and predict the grounding results.

for all objects are extracted by the object encoder (e.g. PointNet++ [23]) and a linear mapping layer, given as:

$$x_i = \mathrm{LN}(W_x \cdot \mathrm{PointNet{+}{+}}(pc_i)), \tag{5}$$

where x_i is the point cloud feature of object i, W_x is a projection matrix. $\mathrm{LN}(\cdot)$ is layer normalization [2].

Multi-View Positional Encoding. The object coordinates are represented by box center coordinates $\{b_1, \ldots, b_M\}$, in the form of XYZ. We rotate them to generate the object coordinates under multiple views, formulated as:

$$b_i^j = R(\theta_v^j) \times b_i, \tag{6}$$

where b_i^j is the coordinate of object i under view j. $R(\theta)$ and θ_v^j are defined in Eq. (2) and Eq. (3). Then we obtain the d-dimensional positional encoding $\mathrm{PE}(b_i^j)$ for object i under view j, given as:

$$\mathrm{PE}(b_i^j) = \mathrm{LN}(W_b[b_i^j, r_i]), \tag{7}$$

where $r_i \in \mathbb{R}^1$ is the box size of object i, W_b is a projection matrix mapping the box information to high dimension embedding, $\mathrm{LN}(\cdot)$ is layer normalization [2], and $[\cdot, \cdot]$ is the concatenate operation.

Object Feature Generation. We obtain final object features $O^j = \{o_1^j, \ldots, o_M^j\}$ under view j by adding the positional encoding PE with the point cloud features x, given as:

$$o_i^j = x_i + \mathrm{PE}(b_i^j), \tag{8}$$

where x_i is shared by multiple views, providing visual information, e.g., categories, colors, etc. $\mathrm{PE}(\cdot)$ provides the size and position information for each view.

3.3. Multi-Modal Feature Fusion

The multi-modal features are fused by the language feature L and the object features O under multiple views. Following [14, 25, 33], we adopt a BERT model [9] as the language encoder. Given the query expression Q with k_1 words, we embed them into d-dimensional feature vectors L, given as:

$$\{l_s, l_1, \ldots, l_{k_1}\} = \mathrm{BERT}(\mathrm{Q}), \tag{9}$$

where l_s is the sentence-level feature, l_i is the feature of i-th word. We fine-tune the BERT during training. To enhance the BERT, a text classifier with two FC layers is applied on l_s, which predicts the object category described by Q, and supervised by a cross-entropy loss $\mathcal{L}_{text}$.

We adopt a standard transformer decoder [29] for the multi-modal feature fusion, which is shared by different views. We fuse the multi-modal feature under view j as:

$$F^j = \mathrm{Decoder}(\mathrm{O^j}, \mathrm{L}), \tag{10}$$

where $F^j = \{f_1^j, \ldots, f_M^j\}$ is multi-modal feature of object i under view j. As shown at the bottom of Fig. 3, in each layer of the decoder, object features O^j first pass through a self-attention module [29], where interactions between objects are modeled. Then object features O^j and language features L are fused through a cross-attention module [29].

3.4. Multi-View Aggregation

After the multi-modal feature fusion, we aggregate the multi-view information from $\{f_i^1, \ldots, f_i^N\}$. The aggregation function should be order-independent as in Eq. (4), here we choose the average aggregation by default, given as:

$$g_i = \sum_{j=1}^{N} \frac{f_i^j}{N}, \tag{11}$$

where $G = \{g_1, \ldots, g_M\}$ is the final multi-modal feature of each object. Finally we obtain the grounding score by applying two FC layers on G.

We aggregate the multi-view information after the multi-modal fusion by default and then make the final prediction. We can also aggregate the multi-view information at earlier stages (e.g., after object feature encoding). There are also other alternatives to the aggregation function (e.g. $\max$, $\text{avg} + \max$). We explore the performance of different aggregation stages and different aggregation functions in Tab. 6.

3.5. Improving Object Encoder

To enhance the object encoder, we introduce an auxiliary task to learn the classification of 3D objects like previous works [1,25]. However, these works mainly apply a few FC layers on the object encoder to classify objects, using only the visual data but neglecting any multi-modal information in this visual grounding task. We propose the auxiliary task of language-guided classification by leveraging object category labels as the text supervision.

As shown in the top of Fig. 3, given the set of category label texts (e.g. "armchair", "rocking chair", and "sofa chair"), we first extract text features from category labels following Eq.(9). The d-dimensional category text features are denoted as $C = \{c_1, \ldots, c_{k_2}\} \in \mathbb{R}^{k_2 \times d}$, where c_i is the sentence-level feature of category i, and k_2 is the category number (e.g. 524 in Nr3D [1] and 607 in Sr3D [1]). Then classification logits $p_i \in \mathbb{R}^{k_2}$ of object i are calculated by the inner product of category text features C and the object feature x_i. We adopt a cross-entropy loss to supervise p_i, denotes as $\mathcal{L}_{obj}$.

3.6. Overall Loss Functions

We use a cross-entropy loss $\mathcal{L}_{ref}$ to supervise the final grounding score, and two auxiliary losses $\mathcal{L}_{text}$ and $\mathcal{L}_{obj}$ mentioned above to enhance the object encoder and language encoder. The total loss is defined as:

$$\mathcal{L}_{total} = \mathcal{L}_{ref} + \alpha(\mathcal{L}_{text} + \mathcal{L}_{obj}), \tag{12}$$

where α controls the ratio of each loss term. We set $\alpha = 0.5$ by default.

4. Experiments

4.1. Datasets

Nr3D. The Nr3D dataset [1] annotates the indoor 3D scene dataset ScanNet [7] with $45,503$ human utterances. There are a total of 707 unique indoor scenes with target objects from 76 fine-grained classes. Each scene contains no more than six distractors (objects in the same class as the target). Two kinds of data splits are used in Referit3D. The "easy" and "hard" splits depend on whether the scene contains more than two distractors. The "view-dep." and "view-indep." splits depend on whether the referring expression is dependent on the speaker's view or not.

Sr3D/Sr3D+. The Sr3D dataset [1] contains $83,572$ utterances that are automatically generated based on a "target-spatial relationship-anchor object" template. The Sr3D+ dataset further expands Sr3D by adding samples without multiple distractors, resulting in a $114,532$ utterances.

ScanRefer. The ScanRef dataset [3] annotates 800 indoor scenes in ScanNet [7] with $51,583$ human utterances, containing $36,665$, $9,508$, and $5,410$ samples in train/val/test sets, respectively. The data can be divided into "Unique" and "Multiple", depending on whether there are objects of the same class as the target in the scene.

4.2. Experimental Setting

Evaluation metric. For datasets in Referit3D [1] (e.g. Nr3D, Sr3D, and Sr3D+), the proposals are generated from the ground truth. The models are evaluated by the accuracy, which measures the percentage of successful matches between the predicted proposal and the ground truth proposal. For ScanRefer [3], the proposals are generated from a pre-trained detector. We adopt PointGroup [16] as the detector, following InstanceRefer [36]. The Acc@mIoU is adopted as the evaluation metric, where $m\text{IoU} \in \{0.25, 0.5\}$. Acc@$m$IoU measures the fraction of language queries whose predicted box overlaps the ground truth box with 3D intersection over the union (IoU) higher than m.

Implementation details. We set the dimension d in all transformer layers as 768. We adopt the first three layers of BERT [9] as the text encoder and a four layers transformer decoder [29] for multi-modal feature fusion, which makes the number of parameters comparable to SAT [33]. The text encoder is initialized from the first three layers of $\text{BERT}_{\text{BASE}}$ [29]. The fusion decoder is trained from scratch. The number of views N is set to 4. Model optimization is conducted using Adam [19] optimizer with a batch size of 24. We set an initial learning rate of 0.0005 for the model, and the learning rate of the transformer layer is further multiplied by 0.1. We reduce the learning rate by a multiplicative factor of 0.65 every 10 epochs after 40 epochs for a total of 100 epochs.

Arch.	Extra	Overall	Easy	Hard	View-dep.	View-indep.
Nr3D						
ReferIt3D [1]	None	35.6%±0.7%	43.6%±0.8%	27.9%±0.7%	32.5%±0.7%	37.1%±0.8%
InstanceRefer [36]	None	38.8%±0.4%	46.0%±0.5%	31.8%±0.4%	34.5%±0.6%	41.9%±0.4%
3DVG-Transformer [38]	None	40.8% ± 0.2%	48.5% ± 0.2%	34.8% ± 0.4%	34.8% ± 0.7%	43.7% ± 0.5%
FFL-3DOG [10]	None	41.7%	48.2%	35.0%	37.1%	44.7%
TransRefer3D [14]	None	42.1%±0.2%	48.5%±0.2%	36.0%±0.4%	36.5%±0.6%	44.9%±0.3%
LanguageRefer [25]	None	43.9%	51.0%	36.6%	41.7%	45.0%
SAT [33]	2D assist.	49.2%±0.3%	56.3%±0.5%	42.4%±0.4%	46.9%±0.3%	50.4%±0.3%
Ours	None	**55.1%±0.3%** (+11.2%)	**61.3%±0.4%** (+11.3%)	**49.1%±0.4%** (+12.5%)	**54.3%±0.5%** (+12.6%)	**55.4%±0.3%** (+10.4%)
Nr3D w/ Sr3D						
ReferIt3D [1]	None	37.2%±0.3%	44.0%±0.6%	30.6%±0.3%	33.3%±0.6%	39.1%±0.2%
non-SAT [33]	None	43.9%±0.3%	-	-	-	-
TransRefer3D [14]	None	47.2%±0.3%	55.4%±0.5%	39.3%±0.5%	40.3%±0.4%	50.6%±0.2%
SAT [33]	2D assist.	53.9%±0.2%	61.5%±0.1%	46.7%±0.3%	52.7%±0.7%	54.5%±0.3%
Ours	None	**58.5%±0.2%** (+11.3%)	**65.6%±0.2%** (+10.2%)	**51.6%±0.3%** (+12.3%)	**56.6%±0.3%** (+16.3%)	**59.4%±0.2%** (+8.8%)
Nr3D w/ Sr3D+						
ReferIt3D [1]	None	37.6%±0.4%	45.4%±0.6%	30.0%±0.4%	33.1%±0.5%	39.8%±0.4%
non-SAT [33]	None	45.9%±0.2%	-	-	-	-
TransRefer3D [14]	None	48.0%±0.2%	56.7%±0.4%	39.6%±0.2%	42.5%±0.2%	50.7%±0.4%
SAT [33]	2D assist.	56.5%±0.1%	64.9%±0.2%	48.4%±0.1%	54.4%±0.3%	57.6%±0.1%
Ours	None	**59.5%±0.2%** (+11.5%)	**67.4%±0.4%** (+10.7%)	**52.7%±0.4%** (+13.1%)	**59.1%±0.5%** (+16.6%)	**60.3%±0.1%** (+9.6%)

Table 1. Performance on Nr3D trained with or without Sr3D/Sr3D+. The improvement is calculated in comparison with the best competitor without extra assistance.

Arch.	Extra	Overall	Easy	Hard	View-dep.	View-indep.
ReferIt3D [1]	None	40.8%±0.2%	44.7%±0.1%	31.5%±0.4%	39.2%±1.0%	40.8%±0.1%
InstanceRefer [36]	None	48.0%±0.3%	51.1%±0.2%	40.5%±0.3%	45.4%±0.9%	48.1%±0.3%
3DVG-Transformer [38]	None	51.4% ± 0.1%	54.2% ± 0.1%	44.9% ± 0.5%	44.6% ± 0.3%	51.7% ± 0.1%
LanguageRefer [25]	None	56.0%	58.9%	49.3%	49.2%	56.3%
TransRefer3D [14]	None	57.4%±0.2%	60.5%±0.2%	50.2%±0.2%	49.9%±0.6%	57.7%±0.2%
SAT † [33]	2D assist.	57.9%	61.2%	50.0%	49.2%	58.3%
Ours	None	**64.5%±0.1%** (+7.1%)	**66.9%±0.1%** (+6.4%)	**58.8%±0.1%** (+8.6%)	**58.4%±0.8%** (+8.5%)	**64.7%±0.1%** (+7.0%)

Table 2. Performance on Sr3D. The improvement is calculated in comparison with the best competitor without extra assistance.

4.3. 3D Visual Grounding Results

Nr3D. Tab. 1 reports the grounding accuracy on Nr3D [1] dataset. Our MVT greatly surpasses all previous methods. Compared with the best competitor [25] using the same training setting, MVT outperforms the state-of-the-art method by $+11.2\%$, from 43.9% to 55.1%. Even when compared with SAT [33], which utilizes extra 2D semantics assisted its training, our MVT still outperforms it by 5.9% absolute value. Specifically, the high accuracy on the "Easy" split (11.3% better than LanguageRefer [25]) reflects the better visual-language alignment of our MVT. At the same time, the high accuracy on the "View-dep" split (12.6% better than LanguageRefer [25]) reflects the effectiveness and robustness of the multi-view representation learned by our MVT. Tab. 1 also reports the performance of training on Nr3D together with Sr3D and Sr3D+ datasets [1]. Both synthetic datasets expand the scale of the training set, therefore further improving the performance on Nr3D. As the results show, our MVT on "Nr3D w/ Sr3D" and "Nr3D w/ Sr3D+" achieves the overall grounding accuracy of 58.5% and 59.5% respectively, which significantly surpasses the state-of-the-art method [14] by 11.3% and 11.5%.

Sr3D. Tab. 2 shows the grounding accuracy on Sr3D [1]. The MVT achieves great improvements and is significantly better than state-of-the-art methods [14, 33]. Though our approach does not use any additional 2D data, ours is still much better than SAT [33] which is with 2D semantics assisting its training. Compared with TranRefer3D [14] and SAT [33], our MVT exceeds them $+7.1\%$ and $+6.6\%$ respectively.

ScanRefer. Tab. 3 shows the performance on ScanRefer [3]. We follow the same settings with InstanceRefer [36]. Since the detector has completed the 3D object classification and feature extraction steps, we only test the performance of the multi-view representation. As the results show, the overall Acc@0.25 and Acc@0.5 of our

Method	Unique		Multiple		Overall	
	Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5
ScanRefer [3]	65.00%	43.31%	30.63%	19.75%	37.30%	24.32%
TGNN [15]	64.50%	53.01%	27.01%	21.88%	34.29%	27.92%
TGNN [15] + BERT [9]	68.61%	56.80%	29.84%	23.18%	37.37%	29.70%
IntanceRefer [36]	77.45%	**66.83%**	31.27%	24.77%	40.23%	32.93%
Ours (view num = 1)	74.36%	63.04%	29.65%	23.44%	38.33%	31.12%
Ours (view num = 4)	**77.67%**	66.45%	**31.92%**	**25.26%**	**40.80%**	**33.26%**

Table 3. Performance on ScanRefer compared with previous works.

	Decoder	**Aug.**	**Lang-Sup.**	**Multi-View**	Overall	Easy	Hard	View-dep.	View-indep.
(a)					36.9%	43.5%	30.5%	32.9%	38.8%
(b)	√				40.4%	47.5%	33.7%	38.4%	41.4%
(c)	√	√			40.8%	48.4%	33.5%	35.2%↓	43.6%
(d)	√	√	√		46.2%	53.8%	38.9%	42.6%	48.0%
(e)	√	√		√	52.3%	59.1%	45.7%	50.3%↑	53.3%
(f)	√		√	√	53.3%	59.8%	47.0%	54.4%	52.7%
(g)	√	√	√	√	55.1%	61.3%	49.1%	54.3%≈	55.4%

Table 4. Ablation studies of the MVT components on Nr3D.

MVT with the view number of 1 is 38.33% and 31.12%, respectively. After increasing the view number to be 4, the performance is improved to 40.80% and 33.26% respectively.

Inference Time. On a TITAN X (Pascal), for one data pair of a 3D scene and a language description, the average inference time of MVT with view number $N = 4$ is 264ms, and the inference time with $N = 1$ is 261ms. The increase in inference time by our multi-view modeling is slight.

4.4. Ablation studies

4.4.1 Effectiveness of each Component

To investigate the effectiveness of each component in MVT, we conduct detailed ablation studies on Nr3D [1], shown in Tab. 4. The baseline model in row (a) simply passes object features through several linear layers and then fuses object features with language features through multiplication directly. Thereby the baseline model does not model the relationship between objects. After we introduce the transformer decoder [29] to model object relationship and vision-language fusion in row (b), the performance is improved from 36.9% to 40.4%. We add random rotation augmentation $R(\theta_{aug})$ to the input scene in row (c), where θ_{aug} is randomly sampled from θ_v in Eq.(3) with $N = 4$. The improvement in the "Easy" split shows that the augmentation improves the performance of the object encoder. However, the rotation augmentation cannot eliminate the dependence on the specific views, and the performance in the "View-dep" split degraded from 38.4% to 35.2%. Row (d) and (e) bear out the effectiveness of the proposed language-guided supervision and multi-view modeling, respectively. They improve the performance to 46.2% and 52.3% respectively. The improvement of language-guided supervision comes from better vision-language alignments. The giant jump of the "View-dep." split from 35.2% to 49.1% shows that our proposed multi-view space modeling effectively learns a view-robust representation, eliminating the dependence on specific views. Adopting both language-guided supervision and multi-view encoding in (g) can improve the overall accuracy to 55.1%, surpassing all previous methods. We also compare the performance w/ and w/o rotation augmentation after adding the multi-view modeling. Comparing the performance in row (f) and row (g), we find that after adding the multi-view modeling, the rotation augmentation can enhance the object encoder and cause no degradation in the "View-dep." split.

View Number Train	View Number Test	Overall	Easy	Hard	View-dep.	View-indep.
4	1	51.6%	58.4%	45.0%	50.7%	52.0%
4	2	54.3%	60.6%	48.3%	53.7%	54.7%
4	4	55.1%	61.3%	49.1%	54.3%	55.4%
4	8	54.8%	61.1%	48.8%	54.2%	55.1%
1	1	46.2%	53.8%	38.9%	42.6%	48.0%
2	2	51.9%	60.2%	43.9%	50.7%	52.5%
8	8	53.4%	60.9%	46.2%	53.5%	53.4%

Table 5. Ablation of view numbers on Nr3D.

4.4.2 Effectiveness of Multi-View Modeling.

Number of views. Tab. 5 shows the performance with different numbers of view on Nr3D [1]. We first compare the performance of 4-view training but test in different numbers of views, which shows that the improvement of MVT comes from two parts. Firstly, when training with 4 views but testing with 1 view, the overall accuracy 51.6% is much higher than training in 1 view only, i.e., 46.2%, and the "View-dep." accuracy also increases by 8.1%. This result shows that multi-view modeling can learn an effective representation that can benefit different views. Secondly, when increasing the view number during

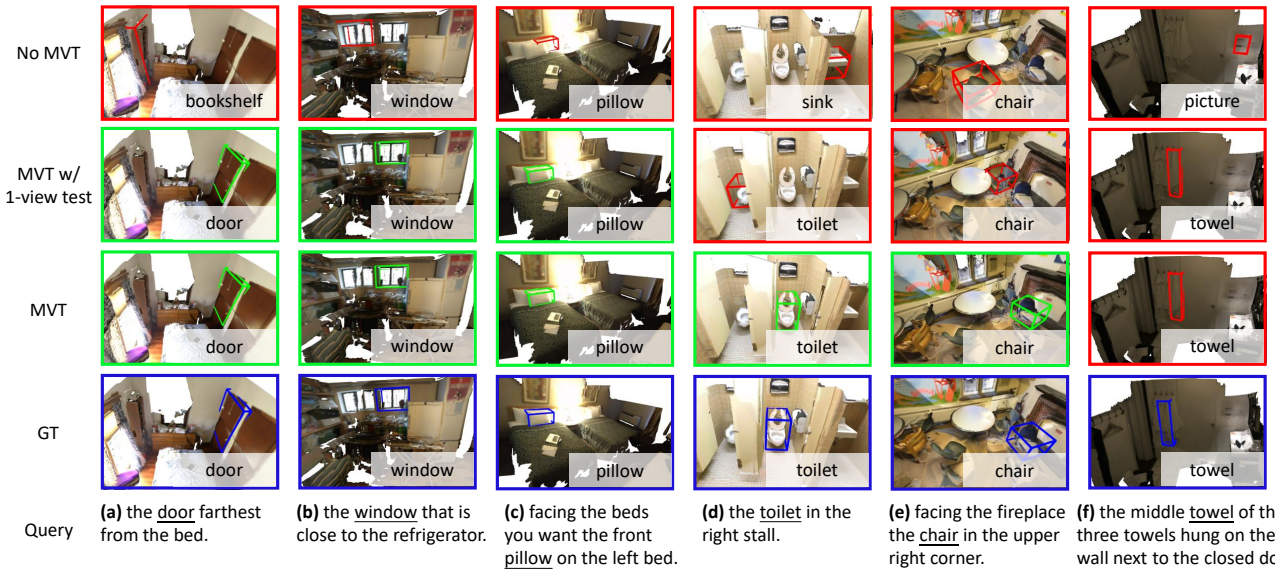

Figure 4. The visualization results. The green/red/blue colors indicate the correct/incorrect/ground truth boxes. Best viewed in color.

testing, the grounding accuracy can still be improved. This observation tells that the information aggregation from the multi-view space can be helpful. We also compare the results of training with other view numbers. When training with only 2 views, the model's performance can achieve 51.9%. When two views are relatively close, the position information provided by them for visual grounding is similar and redundant, thus we found 4-view can already provide a robust multi-view representation. When training with more views (e.g., $N = 8$ views), redundant views cannot improve performance continually and may cause a little performance degradation due to low training efficiency.

Aggregation	Overall	Easy	Hard	View-dep.	View-indep.
After PE	42.5%	49.5%	35.8%	37.4%	45.0%
After $+x$	47.4%	54.2%	41.0%	44.3%	49.0%
After $*L$	55.1%	61.3%	49.1%	54.3%	55.4%
Avg	55.1%	61.3%	49.1%	54.3%	55.4%
Max	52.7%	60.0%	45.7%	51.7%	53.2%
Avg + Max	55.2%	62.1%	48.5%	54.4%	55.6%

Table 6. Ablation of multi-view aggregation on Nr3D.

Multi-view aggregation. Tab. 6 shows performances of different multi-view aggregation settings. We first adopt average aggregation and test different aggregation stages. The positional encoding is just a linear mapping with LayerNorm [2]. If we aggregate after the positional encoding generation (PE), it is equivalent to averaging the coordinates of multiple views. It performs badly since the position information is destroyed. To aggregate after object feature generation ($+x$), we first pass the object features into a 2-layers transformer encoder [29] and then fuse object features from different views. It keeps the position information but still can not achieve the best performance. Aggregation after multi-modal feature fusion ($*L$) achieves the best performance. We also test different aggregation functions. "Avg" is more effective than "Max", and "Avg + Max" achieves similar performance as "Avg".

Visualization. We show the visualization results in Fig. 4. "No MVT" is the baseline model same as in Tab. 4 (b), "MVT w/ 1-view test" is the MVT model but test under 1-view, "MVT" is our proposed model. From (a, d, f), we can see that the baseline model's object recognition ability is relatively poor, and it is easy to predict objects of the wrong categories. By learning a view-robust representation, MVT can effectively learn the position correspondence between 3D scenes and language queries, thus predicting correctly in (b, c). We can see from (d, e) that information aggregation from the multi-view space can achieve better performance. We also show the failure cases of MVT in (f). When faced with complicated language queries and spatial relationships, MVT may also predict objects of the wrong categories or in the wrong positions.

5. Conclusion

In this paper, we propose a Multi-View Transformer (MVT) for 3D visual grounding. In MVT, the given 3D scene is projected to a multi-view space, in which the position information of the 3D scene under different views are modeled simultaneously. Based on it, MVT learns a more robust multi-modal representation for 3D visual grounding, which eliminates the dependence on specific views. Extensive experiments show that our MVT outperforms all state-of-the-art methods by a large margin. Detailed ablation studies show the effectiveness of all components in our model.

References

[1] 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 *ECCV*, 2020. 1, 2, 3, 5, 6, 7

[2] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv*, 2016. 4, 8

[3] Dave Zhenyu Chen, Angel X Chang, and Matthias Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. In *ECCV*, 2020. 1, 2, 3, 5, 6, 7

[4] Xinlei Chen, Hao Fang, Tsung-Yi Lin, Ramakrishna Vedantam, Saurabh Gupta, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco captions: Data collection and evaluation server. *arXiv*, 2015. 2

[5] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In *CVPR*, 2017. 3

[6] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *NIPS*, 2014. 2

[7] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017. 2, 5

[8] Karan Desai and Justin Johnson. Virtex: Learning visual representations from textual annotations. In *CVPR*, 2021. 2

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, 2019. 4, 5, 7

[10] 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. *arXiv*, 2021. 2, 6

[11] Qi Feng, Vitaly Ablavsky, and Stan Sclaroff. Cityflow-nl: Tracking and retrieval of vehicles at city scale by natural language descriptions. *arXiv*, 2021. 1

[12] Andreas Geiger, Philip Lenz, and Raquel Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *CVPR*, 2012. 3

[13] Lluis Gomez, Yash Patel, Marçal Rusiñol, Dimosthenis Karatzas, and CV Jawahar. Self-supervised learning of visual features through embedding images into text topic spaces. In *CVPR*, 2017. 2

[14] 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*, 2021. 1, 2, 4, 6

[15] Pin-Hao Huang, Han-Hung Lee, Hwann-Tzong Chen, and Tyng-Luh Liu. Text-guided graph neural networks for referring 3d instance segmentation. In *AAAI*, 2021. 7

[16] Li Jiang, Hengshuang Zhao, Shaoshuai Shi, Shu Liu, Chi-Wing Fu, and Jiaya Jia. Pointgroup: Dual-set point grouping for 3d instance segmentation. In *CVPR*, 2020. 2, 5

[17] Armand Joulin, Laurens Van Der Maaten, Allan Jabri, and Nicolas Vasilache. Learning visual features from large weakly supervised data. In *ECCV*, 2016. 2

[18] Sahar Kazemzadeh, Vicente Ordonez, Mark Matten, and Tamara Berg. Referitgame: Referring to objects in photographs of natural scenes. In *EMNLP*, 2014. 1

[19] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 5

[20] Daqing Liu, Hanwang Zhang, Feng Wu, and Zheng-Jun Zha. Learning to assemble neural module tree networks for visual grounding. In *ICCV*, 2019. 1

[21] Vivek Mittal. Attngrounder: Talking to cars with attention. In *ECCV*, 2020. 1

[22] Bryan A Plummer, Liwei Wang, Chris M Cervantes, Juan C Caicedo, Julia Hockenmaier, and Svetlana Lazebnik. Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *ICCV*, 2015. 1

[23] Charles R Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *arXiv*, 2017. 4

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv*, 2021. 2

[25] Junha Roh, Karthik Desingh, Ali Farhadi, and Dieter Fox. Languagerefer: Spatial-language model for 3d visual grounding. In *CoRL*, 2021. 1, 2, 4, 5, 6

[26] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 1

[27] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 2008. 2

[28] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *ICCV*, 2015. 3

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, 2017. 2, 4, 5, 7, 8

[30] Liwei Wang, Jing Huang, Yin Li, Kun Xu, Zhengyuan Yang, and Dong Yu. Improving weakly supervised visual grounding by contrastive knowledge distillation. In *CVPR*, 2021. 1

[31] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *CVPR*, 2019. 1

[32] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson env: Real-world perception for embodied agents. In *CVPR*, 2018. 1

[33] Zhengyuan Yang, Songyang Zhang, Liwei Wang, and Jiebo Luo. Sat: 2d semantics assisted training for 3d visual grounding. In *ICCV*, 2021. 1, 2, 4, 5, 6

[34] Licheng Yu, Zhe Lin, Xiaohui Shen, Jimei Yang, Xin Lu, Mohit Bansal, and Tamara L Berg. Mattnet: Modular attention network for referring expression comprehension. In *CVPR*, 2018. 1

[35] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *ECCV*, 2016. 1

[36] 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 *ICCV*, 2021. 1, 2, 5, 6, 7

[37] Yuhao Zhang, Hang Jiang, Yasuhide Miura, Christopher D Manning, and Curtis P Langlotz. Contrastive learning of medical visual representations from paired images and text. *arXiv*, 2020. 2

[38] Lichen Zhao, Daigang Cai, Lu Sheng, and Dong Xu. 3dvg-transformer: Relation modeling for visual grounding on point clouds. In *ICCV*, 2021. 2, 6

[39] Fengda Zhu, Yi Zhu, Xiaojun Chang, and Xiaodan Liang. Vision-language navigation with self-supervised auxiliary reasoning tasks. In *CVPR*, 2020. 1