ViewSRD: 3D Visual Grounding via Structured Multi-View Decomposition (Supplementary Material)

Ronggang Huang[1*], Haoxin Yang[1*†], Yan Cai[1],
Xuemiao Xu[12345†], Huaidong Zhang[1], Shengfeng He[6].

[1] South China University of Technology. [2] Guangdong Engineering Center for Large Model and GenAI Technology.
[3] State Key Laboratory of Subtropical Building and Urban Science. [4] Ministry of Education Key Laboratory of Big Data and Intelligent Robot.
[5] Guangdong Provincial Key Lab of Computational Intelligence and Cyberspace Information. [6] Singapore Management University.

A. Summary

This supplementary material offers detailed technical information on key components of our proposed framework, which were abbreviated in the main text. Specifically, we elaborate on the following: the target classifier $Clas$, the examples in template and the details of the sentence matching algorithm in Section 3.1; details of the loss function in Section 3.5.

B. Target Classifier $Clas$

The target label classifier, denoted as $Clas$, is designed to regress the target's class label from an input query Q. By determining the category of Q, we then determine which word in Q belongs to that category to determine which word to use as the *Target*. Essentially, $Clas$ processes an input sentence by first encoding it via a pre-trained BERT [2] model to obtain a semantic embedding, and then maps this embedding through a multi-layer perceptron (MLP) to yield a probability distribution over C target classes.

Specifically, given an input query Q, we first compute its embedding:

$$F_Q = \mathrm{BERT}(Q), \tag{1}$$

where $F_Q \in \mathbb{R}^{K\times d}$ represents the BERT output for the query Q, with K being the number of tokens in Q and d denoting the feature dimension. We select the embedding corresponding to the [CLS] token as the aggregate representation h. This h is then fed into the MLP-based classification head of $Clas$, implemented as a fully connected layer that produces logits $z \in \mathbb{R}^C$. These logits are subsequently converted into a probability distribution over the C classes via the softmax function:

$$P(c \mid Q) = \frac{\exp(z_c)}{\sum_{j=1}^{C}\exp(z_j)}, \tag{2}$$

where z_c denotes the c-th element of z.

The classifier $Clas$ is trained by minimizing the standard cross-entropy loss:

$$\mathcal{L}_{\mathrm{cls}} = -\sum_{c=1}^{C} y_c \log\big(P(c \mid Q)\big), \tag{3}$$

where y_c is the one-hot encoded ground-truth label for class c. During inference, the predicted target label is given by:

$$T_{lab} = \arg\max_{c}\ P(c \mid Q). \tag{4}$$

In summary, $Clas$ effectively bridges the input query and the target object label by leveraging BERT-based encoding followed by MLP-based classification, ensuring robust target label regression in language-conditioned 3D grounding.

C. Examples in SRD Module's Template

In our Simple Relation Decoupling (SRD) module, different examples help LLMs in comprehending various descriptions of spatial relationships. To emphasize those relationships that are particularly sensitive to viewpoint changes, we select spatial relations whose interpretation can vary significantly with the observer's perspective. All examples we used are as follows:

- The {*target*} is close to the {*anchor*}.
- The {*target*} is far away from the {*anchor*}.
- The {*target*} is on the left of the {*anchor*}.
- The {*target*} is on the right of the {*anchor*}.
- The {*target*} is in front of the {*anchor*}.
- The {*target*} is behind the {*anchor*}.

D. Sentence Matching Algorithm

As mentioned in Section C, we predefine an Example Set E, which consists of k examples in total. Given an anchor, it returns k sentences. We employ a Sentence Matching Algorithm, as outlined in Algorithm 1, to select the sentence that

Algorithm 1 Sentence Matching Algorithm

```
1:  Input: Utterance $U$, Anchor set $O$, Target label $y$
2:  $N_q$ = len(U), $N_{anchor}$ = len(O)
3:  For each anchor in Anchor set $O$ do:
4:      For each example in Example set $E$ do:
5:          query = ApplyTemplate($U$, anchor, example)
6:          sentences = LLM(query)
7:          tokens = Tokenizer($U$, sentences)
8:          Predict labels = $Clas$(token)
9:          $S_{consist}$ = GetConsistScore(Predict labels)
10:         $S_{similar}$ = GetSimilarScore(sentences,U)
11:         Score = ComputeScore($S_{consist}$,$S_{similar}$)
12:         Update the simplified sentence and Max_Score
13:         End for
14:       Anchor_LLMsentence.append(simplified sentence)
15: End for
16: Output: Sentence with Maximum Similarity
```

best aligns semantically with the original sentence. Specifically, after the input of the target, a single anchor, and k examples, we obtained k candidate sentences, denoted as $S = \{S_1, S_2, \ldots, S_k\}$, and the original complex query is denoted as S_q through LLMs. All sentences are then predicted through the Target Classifier $Clas$, to obtain pseudo-label $\hat{P}_q$ and $\hat{P}_i, i \in \{1, 2, \ldots, k\}$. The matching score consists of two components. The first is the consistency score, which is the evaluation score for maintaining consistency of the pseudo-label between the generated sentence and the query. The corresponding $S_{consist}$ is formulated as follows:

$$S_{consist} = \begin{cases} 1, & \text{if } \hat{P}_q = \hat{P}_i, \\ 0, & \text{otherwise.} \end{cases} \tag{5}$$

To preserve the semantic integrity of the generated sentence while preventing redundant generation from LLMs, we introduce the second part of the evaluation score: the similarity score. This score primarily considers the shared word count and the lengths of the sentences. We break down both sentences into individual words and calculate the number of shared words N_{co_word} between S_i and S_q. To counteract the advantage of longer sentences in similarity calculation, we employed TextRank Algorithm[4] to compute the sentence similarity, denoted as $S_{Similar}$. Finally, the similarity score is calculated by dividing the shared word count by the sum of the logarithms of the lengths of the two sentences. The formula is as follows:

$$S_{Similar} = \frac{N_{co\_word}}{\log(N_q) + \log(N_s)}, \tag{6}$$

where N_q and N_s denote the length of the query and the generated sentence. In addition, we have defined an ideal decoupling sentence length N^* based on the length of the sentence and the number of anchors in the sentence. Using N^* as the center, the corresponding weights are obtained with a peak value 1, gradually decaying towards both ends. For instance, if N_q is 7 and there are 2 anchors, the weight probability distribution would be {0.4, 0.6, 0.8, 1, 0.8, 0.6, 0.4}. When $N_s = 2$, the corresponding weight probability w is 0.6. Therefore, the corresponding formula for the score is as follows:

$$\text{Score} = w \cdot S_{Similar} + S_{consist}. \tag{7}$$

E. Loss Function

Object-level Loss $\mathcal{L}_{\mathbf{Object}}$: Followed by MVT[3], semantic class representations $\{L_c\}$ are obtained by encoding class name tokens with a pre-trained language encoder and extracting the [CLS] token. Given projected object features $F' \in \mathbb{R}^{N\times D}$ (for N objects with feature dimension D) and class representations $L \in \mathbb{R}^{C\times D}$, we compute the predicted logits as

$$P_{\text{obj}} = F' \cdot L^T \in \mathbb{R}^{N\times C}. \tag{8}$$

The ground truth is represented as a one-hot encoded matrix $T_{\text{obj}} \in \mathbb{R}^{N\times C}$. The object-level loss is then defined as

$$\mathcal{L}_{\text{Object}} = -\sum_{j=1}^{N}\sum_{c=1}^{C} T_{\text{obj}}(j,c)\, \log\Big(\text{softmax}(P_{\text{obj}})_{jc}\Big). \tag{9}$$

This loss ensures that the object features capture critical geometric properties, such as shape and center, and align accurately with their semantic categories.

Referential Loss $\mathcal{L}^P_{\mathbf{ref}}$: To supervise spatial alignment, followed by CoT3DRef[1], , we obtain in parallel the anchor position labels $\mathbf{T}_{\text{anchor}} \in \mathbb{R}^I$ and the target position label $\mathbf{T}_{\text{target}} \in \mathbb{R}^1$, and concatenate them to form the referential ground truth

$$\mathbf{T}_{\text{ref}} = \text{concat}\left(\mathbf{T}_{\text{anchor}}, \mathbf{T}_{\text{target}}\right) \in \mathbb{R}^{(I+1)}, \tag{10}$$

where each element $T_{\text{ref},p}$ (for $p = 1, \ldots, I+1$) indicates the correct object index (among the i objects present in the scene) for that spatial position. After view aggregation and prediction head, the model outputs referential logits $L_{\text{ref}} \in \mathbb{R}^{(I+1)\times i}$, where each row corresponds to a spatial position (anchor or target) and each column represents one of the i objects. The referential loss is defined as

$$\mathcal{L}^P_{\text{ref}} = -\frac{1}{I+1}\sum_{p=1}^{I+1} \log \frac{\exp\left(L_{\text{ref},p,T_{\text{ref},p}}\right)}{\sum_{r=1}^{i}\exp\left(L_{\text{ref},p,r}\right)}. \tag{11}$$

This loss encourages the textual module to generate discriminative features that accurately capture the semantic nuances of the grounding statement, effectively distinguishing between the anchor and target labels.

Sentence-level Loss $\mathcal{L}_{\text{Sent}}$: Followed by CoT3DRef[1], we supervise the textual module using a standard cross-entropy loss. Let I denote the number of anchor class labels in the original sentence. For each sample, we form a label sequence by concatenating the I anchor labels with one target label, yielding a sequence of length

$$S = I + 1. \tag{12}$$

After Textual Aggregation, the classification head produces language prediction logits $\mathbf{P}_{\text{sent}} \in \mathbb{R}^{S\times C}$, where C is the total number of classes. The ground truth is represented as a vector $\mathbf{T}_{\text{sent}} \in \mathbb{R}^{S}$, with each element $T_{\text{sent},i}$ indicating the correct class index for the i-th position. The language loss is defined as

$$\mathcal{L}_{\text{Sent}} = -\frac{1}{S}\sum_{i=1}^{S} \log \frac{\exp\left((\mathbf{P}_{\text{sent}})_{i,T_{\text{sent},i}}\right)}{\sum_{j=1}^{C}\exp\left((\mathbf{P}_{\text{sent}})_{ij}\right)}. \tag{13}$$

This loss encourages the textual module to generate discriminative features that accurately capture the semantic nuances necessary for language grounding.

F. t-SNE visualization of CCVTS

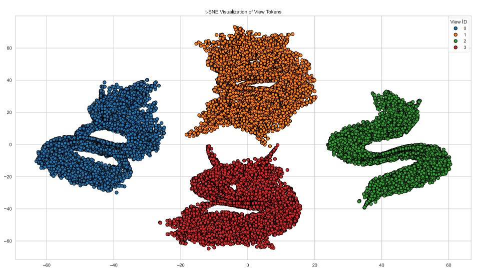

Figure 1. t-sne Visualization of CCVTs.

To further investigate how CCVTs encode perspective information, we visualize the learned features using t-SNE, as shown in Fig. 1. Tokens originating from different viewpoints form well-separated clusters, while those from the same viewpoint are closely grouped. This clear structural separation illustrates that CCVTs effectively capture and preserve view-specific semantics, which are essential for accurate cross-view alignment in 3D visual grounding.

G. Analysis of Loss Weight.

Table 1 presents an ablation study on the weighting scheme of the loss components $\mathcal{L}_{Ref}$, $\mathcal{L}_{Obj}$, and $\mathcal{L}_{Sent}$. The best accuracy of **69.9%** is achieved when the loss weights are set to $\lambda_{Obj} = 1.0$, $\lambda_{Ref} = 0.5$, and $\lambda_{Sent} = 0.5$, confirming the effectiveness of our multi-loss formulation with this specific weighting scheme.

Table 1. Ablation on loss weights.

λ_{Ref}	λ_{Obj}	λ_{Sent}	Accuracy (%)
1.0	1.0	0.0	69.1
1.0	0.0	1.0	63.9
1.0	0.25	0.25	68.4
1.0	0.5	0.5	**69.9**
1.0	0.75	0.75	69.6
1.0	1.0	1.0	69.8

Notably, omitting either $\mathcal{L}_{Obj}$ or $\mathcal{L}_{Sent}$ leads to significant performance degradation (to 63.9% and 69.1%, respectively), highlighting the importance of jointly modeling object-level and sentence-level supervision. We also observe that moderate weighting (e.g., 0.25 or 0.5) for $\mathcal{L}_{Obj}$ and $\mathcal{L}_{Sent}$ improves over their absence, but still underperforms compared to the balanced setting. These results validate that all three components contribute synergistically to grounding performance.

References

[1] Eslam Mohamed Bakr, Mohamed Ayman, Mahmoud Ahmed, Habib Slim, and Mohamed Elhoseiny. Cot3dref: Chain-of-thoughts data-efficient 3d visual grounding. *The Twelfth International Conference on Learning Representations*, 2024. 2, 3

[2] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[3] Shijia Huang, Yilun Chen, Jiaya Jia, and Liwei Wang. Multi-view transformer for 3d visual grounding. In *CVPR*, pages 15524–15533, 2022. 2

[4] Rada Mihalcea and Paul Tarau. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, pages 404–411, 2004. 2