A. Qualitative Study

A.1. Coarse-to-fine Visualization

Figure 4 shows the visualization results of the coarse-to-fine semantic matching pipeline. Given the input point cloud, we first extract $M_{\mathrm{p}} = 256$ proposals by the 3D object detector. (When doing visualization, we employ the Non-Maximum Supperssion[27] algorithm on these proposals to filter out some duplicate or overlapping boxes.) Then we conduct the coarse-grained candidate selection among these proposals to get $K = 8$ candidates. These candidates do the fine-grained semantic matching with the sentence query one by one to finally produce the best-match result.

In all three cases, the coarse-grained module effectively selects 8 candidates related to the target object (mostly belonging to the target category). For example, in case (b), the sentence asks for a black chair near a table and facing the wall. Firstly, the module filters out objects (*tables*, *doors*) that do not belong to the target category (*chair*). Secondly, with the assistance of the feature similarity matrix, the module's selection is also consistent with the sentence description ("by the table" and "facing the wall") to some degree.

After coarse-grained selection, the reconstruction-based semantic matching process aims to differentiate these candidates in a fine-grained way. We get the correct answer in both cases (a) and (b), but we fail in case (c). Actually, in case (c), the candidates we get do not contain the target object. Considering that there are 32 different chairs in this scene, it's very likely that the 8 candidates miss the target object since we are not expecting the coarse-grained part to have a deep understanding of the objects in the same category (*chair*). Even in such a condition, the fine-grained part still gets the best-match *chair* ("near the doors", "on the left side", and "near the center") among the 8 candidates, which demonstrates our model's strong ability in semantic matching.

B. Implementation Details

B.1. Model Setting

In this section, we give a detailed description of our model setting. We consider the whole two-stage 3DVG pipeline as off-the-shelf modules. In our practice, we use the pre-trained GroupFree[22] model as the 3D object detector, which contains a PointNet++[30] backbone, 6 layers of transformer decoder and the proposal predict head. The proposal predict head outputs the bounding boxes and the class predictions of $M_{\mathrm{p}} = 256$ object proposals. Then we employ the same attribute encoder, textual encoder and predict module as the supervised 3DJCG[3] model. The attribute encoder aggregates the attribute features (27-dimensional box center and corner coordinates and the 128-dimensional multi-view RGB features) and the initial object features (produced by the object detector) with 2-layer multi-head self-attention modules. The input sentence query is firstly encoded by a GloVe module, and then input to a GRU cell, which produces the text feature. The predict module is simply a 1-layer multi-head cross-attention module between the text feature (Key & Value) and the object features (Query). For the reconstruct module, We mask the input sentence with a random ratio $p = 0.3$. We use NLTK to parse the sentences, and the verbs and nouns are treated as important words. The core of the reconstruct module is a 3-layer transformer decoder. The input point number N_{p}, the proposal number M_{p}, and the candidate number K are set to 50000, 256 and 8, respectively. The dimension of all hidden layers is 288. When calculating the candidates' rewards, we find that instead of reducing from one to zero linearly in steps of $1/(K-1)$, applying a square operation over them is better for increasing the discrimination between the optimal and suboptimal candidates.

B.2. Training and Inference

We follow the weakly supervised setting where none of the object-sentence and bounding box annotations are used during training. We follow [40] to use 8 sentence queries for each scene to accelerate the training process. It takes 20 epochs to train our framework with a batch size of 12 ($i.e.$ there are 96 sentence queries from 12 point clouds in each batch). For the stability of the reconstruction module, we start its training at the second epoch and ignore the highest $K/2$ reconstruction losses after the third epoch. The learning rate is set to 1e-3 with cosine annealing strategy. We employ AdamW optimizer[23] with the weight decay of 5e-4. The hyper-parameters λ_1, λ_2 and λ_3 are set to 2, 2 and 1, respectively.

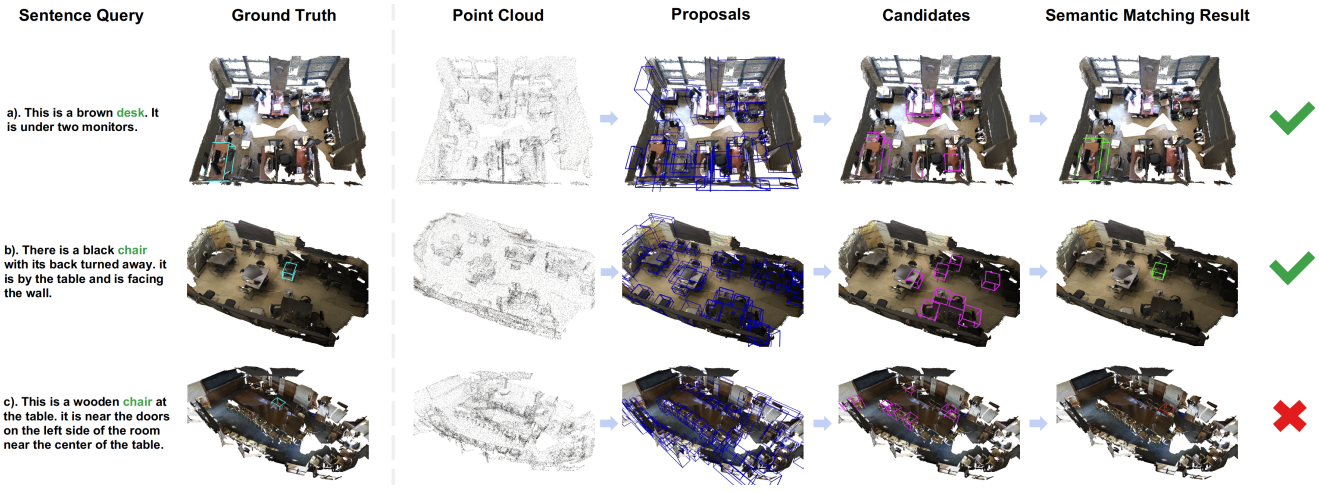

Figure 4. Visualization of the coarse-to-fine semantic matching process. With knowledge distillation, we can directly get matching result from proposals during inference (skipping the time-consuming coarse-to-fine matching).