g3D-LF: Generalizable 3D-Language Feature Fields for Embodied Tasks
Supplementary Material

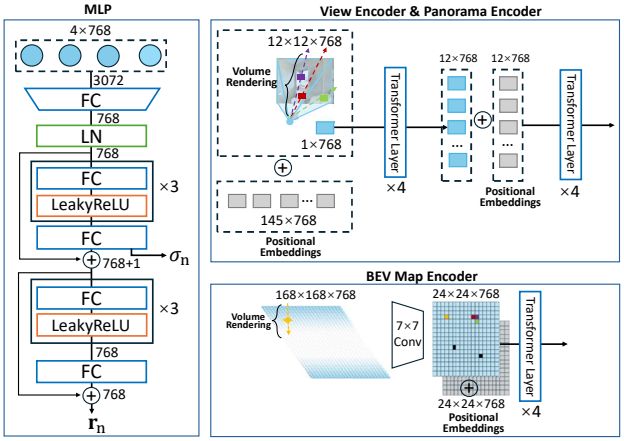

Figure 6. Architecture of modules in the g3D-LF model. FC denotes a fully connected layer, LN denotes layer normalization and LeakyReLU [38] is the activation function.

```
def focal_loss(self, inputs, targets, focal_rate=0.1, focal_weight=1.):
    ce_loss = F.cross_entropy(inputs, targets, reduction='none')
    focal_num = max(int(focal_rate * targets.shape[-1]),1)
    focal_loss = ce_loss.mean() + torch.topk(ce_loss.view(-
    1),focal_num)[0].mean() * focal_weight
    return focal_loss


def sim_matrix_cross_entropy(self, sim_matrix):
    logpt = F.log_softmax(sim_matrix, dim=-1)
    logpt = torch.diag(logpt)
    nce_loss = -logpt
    sim_loss = nce_loss.mean()
    return sim_loss

def contrastive_loss(self, fts_1, fts_2, logit_scale=10.):
    sim_matrix = logit_scale * torch.matmul(fts_1, fts_2.t())
    sim_loss1 = self.sim_matrix_cross_entropy(sim_matrix)
    sim_loss2 = self.sim_matrix_cross_entropy(sim_matrix.T)
    sim_loss = (sim_loss1 + sim_loss2)
    return sim_loss
```

Figure 7. PyTorch implementation of loss functions for the balanced object semantic alignment and the CLIP knowledge distillation.

A. More Details of the g3D-LF Model

Model structure. Figure 6 illustrates the structure of main modules in the g3D-LF model. Compared to HNR [57], g3D-LF improve the MLP network for volume rendering by adding residual connections and replacing ReLU with LeakyReLU, which helps alleviate gradient explosion and neuron death issues during HNR training. Since the number of k-nearest features is set to 4 and the dimension of each aggregated feature is 768, the input dimension of both $\mathbf{MLP}_{view}$ and $\mathbf{MLP}_{BEV}$ networks is 3072. As shown in Figure 6, all transformer-based encoders consist of four-layer transformers.

Settings of novel view prediction. For each sampled point in the rendered ray, we set the search radius for k-nearest features as 0.5 meter. Using *sparse sampling* [57], if no nearby feature points are found within a sampled point's search radius, the latent feature and volume density are set to zero. The rendered ray is uniformly sampled from 0 to 10 meters, and the number of sampled points is set as 501. After volume rendering, the number of rays within a novel view is set as 12×12.

Settings of BEV map prediction. The search radius for k-nearest features is set as 0.4 meter. The rendered ray is uniformly sampled from 0 to 1.6 meters (*i.e.*, vertically from the camera's position to bottom), and the number of sampled points is set as 17. After volume rendering, the number of rays within a BEV map is set as 168×168.

Loss functions. As illustrated in Figure 7 and 8, we present the code for the primary loss functions used in g3D-LF pretraining to provide further details. During training, we apply constant coefficients to balance the contributions of each loss, ensuring they remain within the same order of magnitude.

B. Visualization of the Training Data

As shown in Figure 9, we present a 3D scene from our dataset along with some associated language annotations (scene 00800-TEEsavR23oF from HM3D [46]). The instance-level point cloud precisely annotates instances within the 3D scene, allowing retrieval of language annotations for any position by calculating its neighboring instance points and using the instance IDs.

C. Visualization of the g3D-LF model

As shown in Figure 10 and 11, the g3D-LF model query targets with language on the BEV map. In Figure 10, the left side of each example shows the position of the ground-truth target, while the right side displays the result of querying objects on rays of the BEV map during navigation. The BEV map accurately recognizes both large objects, like *window* and *sofa*, and smaller objects, like *table lamp* and *tap*, by calculating the cosine similarity between ray representations and target text features.

In Figure 11, the left side of each example shows the position of the objects, the middle is the ground-truth position of the long text that contains the target object, while the right side displays the result of querying the long text on the BEV map during navigation. In the 3D scene, multiple objects of the same category often appear. With the excellent ability to understand long texts, our g3D-LF model can achieve more fine-grained long-text queries, distinguishing different instances of the same object category.

```
def fine_grained_contrastive_loss(self, batch_visual_fts, batch_text_fts, logit_scale=10.):
        batch_visual_fts = batch_visual_fts / (torch.linalg.norm(batch_visual_fts, dim=-1, keepdim=True) + 1e-7)
        batch_sim_score = []
        for batch_id in range(len(batch_text_fts)):
            text_fts = batch_text_fts[batch_id]
            text_fts = text_fts[torch.abs(text_fts).sum(-1) != 0]
            text_fts_length = text_fts.shape[0]
            text_fts = text_fts / torch.linalg.norm(text_fts, dim=-1, keepdim=True)
            sim_matrix = logit_scale * torch.matmul(batch_visual_fts, text_fts.t())
            sim_matrix = sim_matrix.view(batch_visual_fts.shape[0],-1)
            sim_score =  torch.topk(sim_matrix,text_fts_length, dim=-1)[0].mean(dim=-1).view(1,-1)
            batch_sim_score.append(sim_score)
        batch_sim_score = torch.cat(batch_sim_score,dim=0)
        sim_loss1 = self.sim_matrix_cross_entropy(batch_sim_score)
        sim_loss2 = self.sim_matrix_cross_entropy(batch_sim_score.T)
        sim_loss = (sim_loss1 + sim_loss2)
        return sim_loss
```

Figure 8. PyTorch implementation of loss function for the fine-grained contrastive learning.

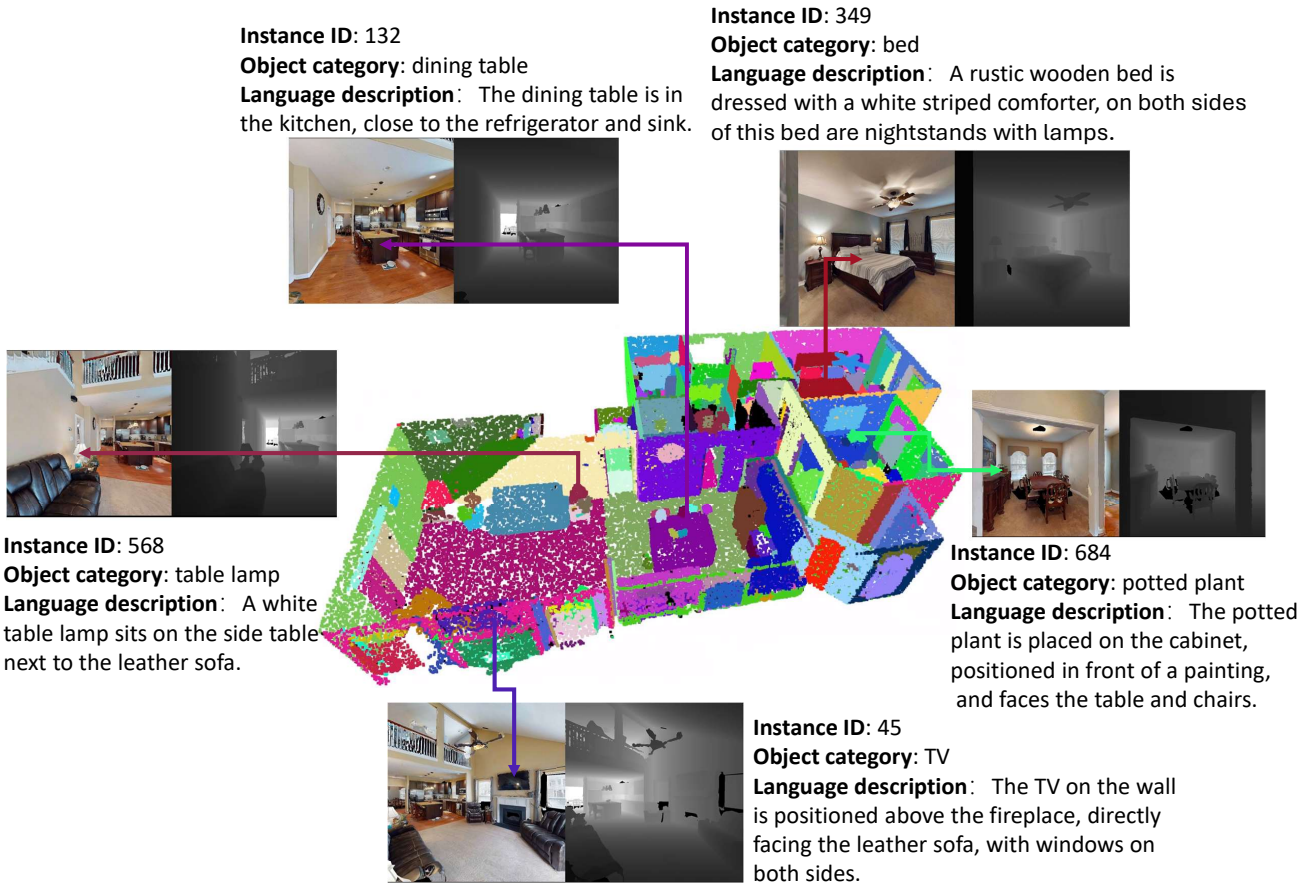

Figure 9. Demonstration of a 3D scene in the training data. Instance-level point clouds mark all instances with object categories, and some instances enriched with language descriptions.

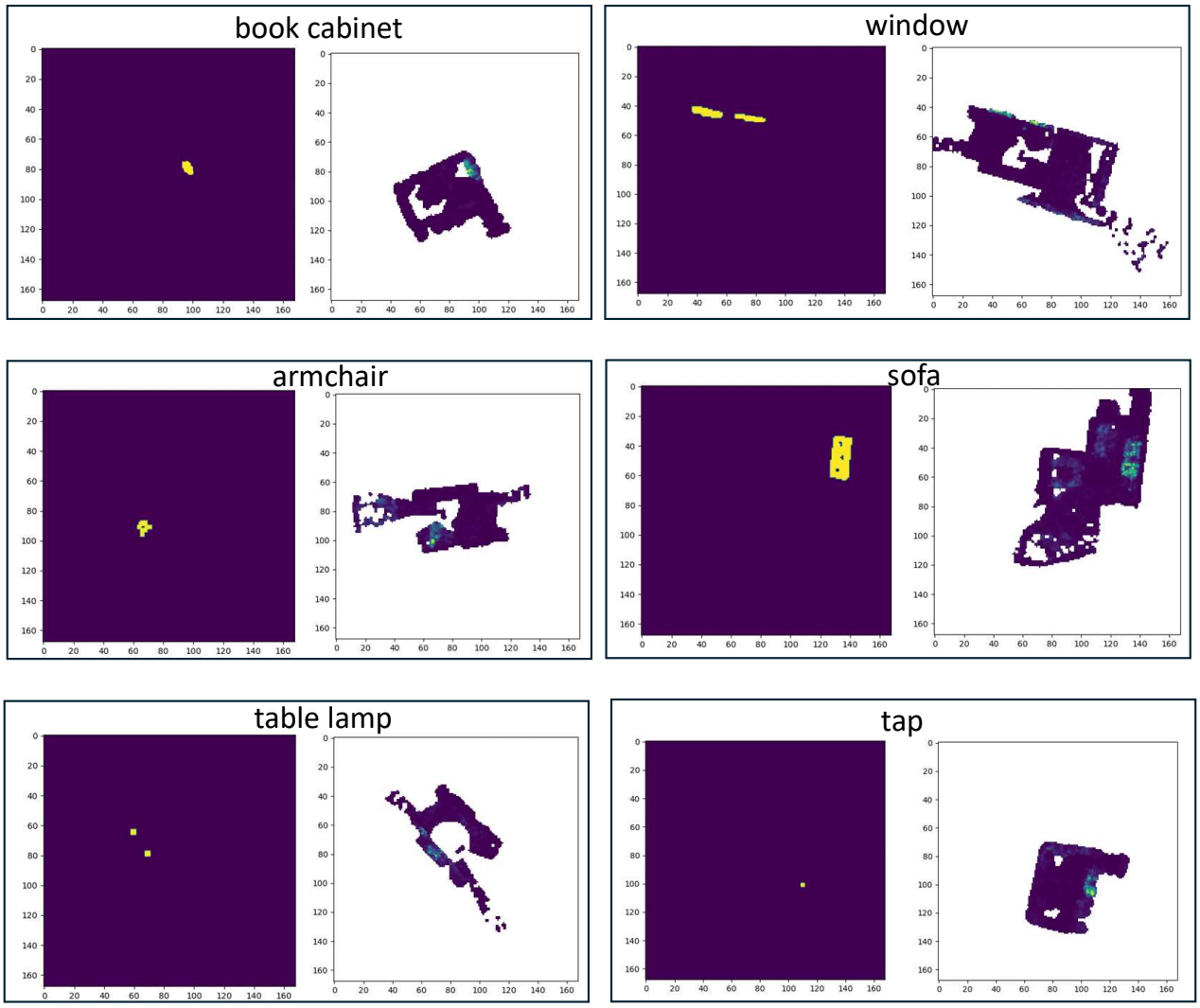

Figure 10. Visualization of querying objects on rays of the g3D-LF's BEV map. The left side of each example is GT, and the right side is the query result. Please zoom in for a better view.

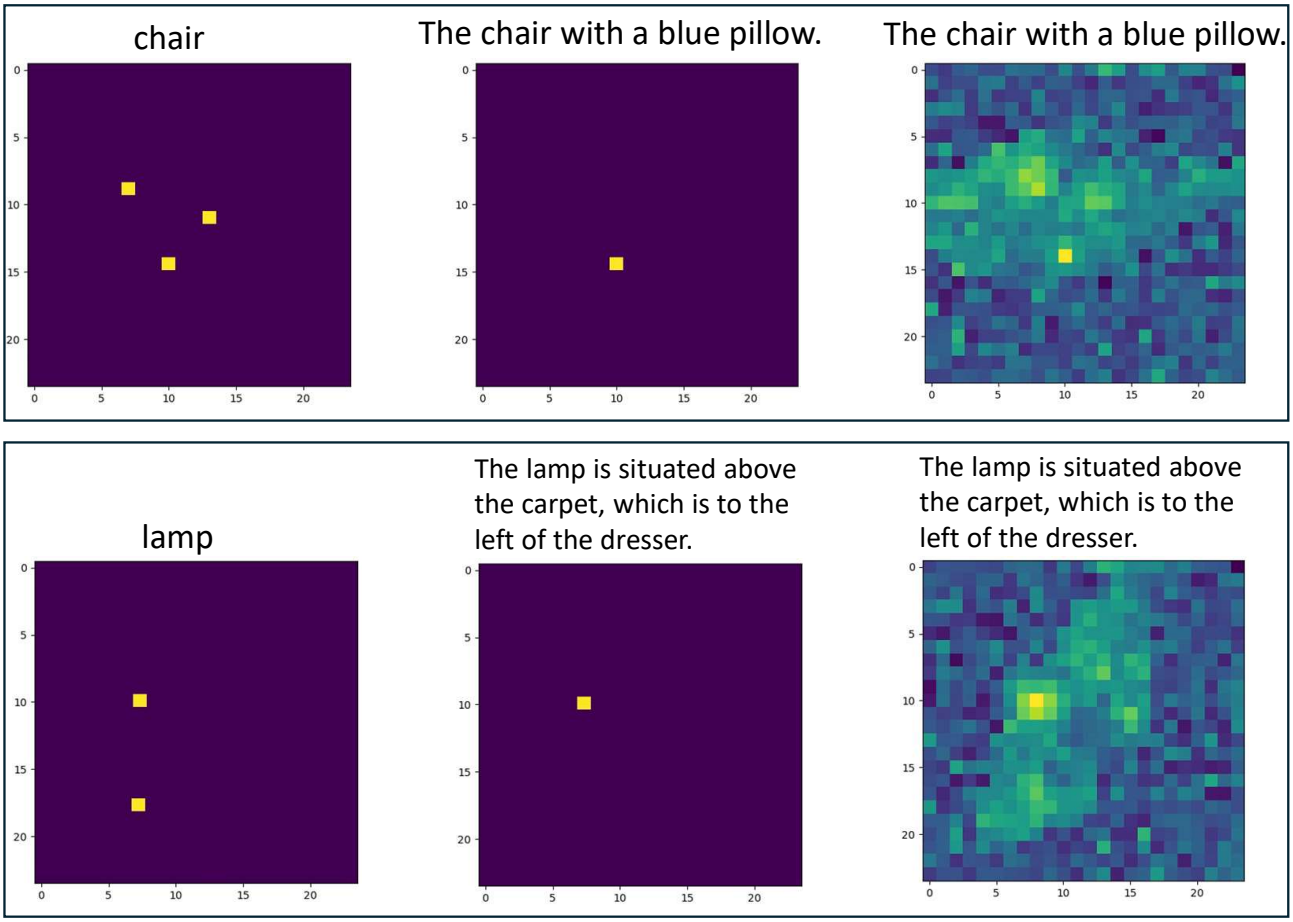

Figure 11. Visualization of querying long texts on the BEV map of our g3D-LF. Each example has the object's GT on the left, the long text GT in the middle, and the query result of the long text on the right. Please zoom in for a better view.