Chain of Semantics Programming in 3D Gaussian Splatting Representation for 3D Vision Grounding

Supplementary Material

Our supplement is organized as follows. In section 6, we provide additional experiments on the ScanRefer dataset. In section 7, we introduce the training details for the 3DGS reconstruction. In section 8, we provide some examples to demonstrate the programming process of our method.

6. ScanRefer Results

We evaluate our proposed method on the ScanRefer dataset, as shown in Table 9.

	Overall		Unique		Multiple	
Methods	Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5	Acc@0.25	Acc@0.5
SAT [39]	44.5	30.1	73.2	50.8	37.6	25.2
MVT [8]	33.3	40.8	77.7	66.5	31.9	25.3
ViL3DRel [9]	47.9	37.7	81.6	68.6	40.3	30.7
3D-VisTA [42]	50.6	45.8	81.6	75.1	43.7	39.1
ZS3DVG [40]	36.4	32.7	63.8	58.4	27.7	24.6
VLM-Grounder [36]	62.4	53.2	87.2	76.6	56.7	47.8
Ours	**61.6**	**55.4**	**79.8**	**71.7**	**54.8**	**49.2**

Table 9. 3D visual grounding results on ScanRefer.

7. Details for 3DGS scene reconstruction training settings

We use the official 3DGS for reconstruction, and all the images used are first cropped 12 pixels from the edges of each image to remove black borders, then resized to 480×640. We trained for 7000 iterations, with sh_degree $= 3$, position_lr_init $= 0.00016$, position_lr_final $= 0.0000016$, feature_lr $= 0.0025$, opacity_lr $= 0.025$, scaling_lr $= 0.005$, self.rotation_lr $= 0.001$, percent_dense = 0.01, densification_interval = 100, $\lambda_{\text{dssim}} = 0.2$ and frame interval = 100. Notably, to align with the masks from the ReferIt3D benchmark, we did not use the densification of 3DGS. Additionally, our method only uses image and pose information without utilizing depth information.

8. Examples for the programming process of our method

Frame Interval	Training Time (7k iter)	Training Time (30k iter)	L1 Loss (7k iter)	PSNR (7k iter)	L1 Loss (30k iter)	PSNR (30k iter)	Model Size (7k/30k iter)
10	2:34	-	0.054	19.16	-	-	273MB / -
20	2:38	18:29	0.047	19.98	0.038	20.92	279MB / 545MB
100	2:27	17:14	0.029	23.17	0.016	29.61	348MB / 611MB
200	2:38	16:50	0.016	31.27	0.011	36.47	347MB / 537MB

Table 10. Frame Interval Analysis Results

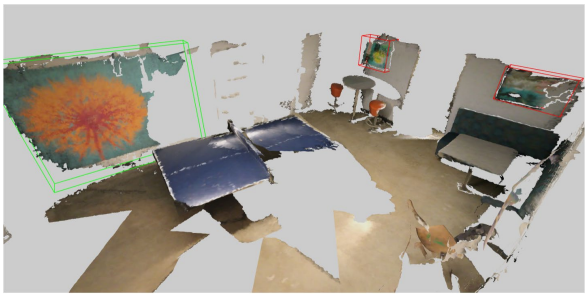
Our method finally locates objects through programming, making the reasoning process relatively transparent compared to end-to-end neural networks. In this section, we provide four examples to demonstrate the reasoning process of our method in scenarios involving spatial relationships, distance between two objects, color, and viewpoint-related information.

Utterance: Choose the table that sits against the wall.

```
import numpy as np

# Define objects based on the input data
objects = {
    '2': {'instance_label': 'table', 'color': [0.37, 0.34, 0.31], 'center': [4.62, 3.53, 0.5], 'size': [1.47,
3.56, 0.86]},
    '14': {'instance_label': 'wall', 'color': [0.61, 0.56, 0.53], 'center': [6.22, 6.76, 1.21], 'size': [0.33,
0.93, 2.33]},
    '16': {'instance_label': 'wall', 'color': [0.69, 0.66, 0.61], 'center': [4.64, 6.85, 1.25], 'size': [3.2,
0.35, 2.38]},
    '17': {'instance_label': 'table', 'color': [0.45, 0.36, 0.28], 'center': [4.94, 6.68, 0.51], 'size': [1.67,
0.43, 0.85]},
    '21': {'instance_label': 'wall', 'color': [0.42, 0.38, 0.34], 'center': [2.2, 3.69, 1.68], 'size': [0.42,
4.81, 1.68]},
    '29': {'instance_label': 'wall', 'color': [0.42, 0.38, 0.33], 'center': [4.1, 0.21, 1.34], 'size': [3.13,
0.2, 2.53]}
}

# Find the table that sits against the wall
selected_table_id = None
selected_wall_id = None
for object_id, object_data in objects.items():
    if object_data['instance_label'] == 'table':
        selected_table_id = object_id
    if object_data['instance_label'] == 'wall':
        wall_center = np.array(object_data['center'])
        wall_position = wall_center[0]
        table_center = np.array(object_data['center'])
        table_position = table_center[0]
        # Condition to determine if the table is against the wall
        if abs(wall_position - table_position) < (object_data['size'][0] + object_data['size'][0]) / 2:
            selected_wall_id = object_id

output_object_id = selected_table_id if selected_wall_id is not None else None

output_object_id
```

Figure 6. Example (a) for our method.

Figure 6 illustrates the process of inferring the spatial relationship between a table and a wall and identifying the ID of a table positioned against a wall based on their spatial coordinates and sizes. Figure 7 presents an example of reasoning with color information (*blue*), identifying the picture beside the blue ping pong table. Figure 8 demonstrates the process of locating objects by calculating the Euclidean distances between multiple objects and grounding the bed closest to the bathroom. Figure 9 demonstrates the reason-



Utterance: The largest picture of a dandelion, closest to the blue ping pong table.

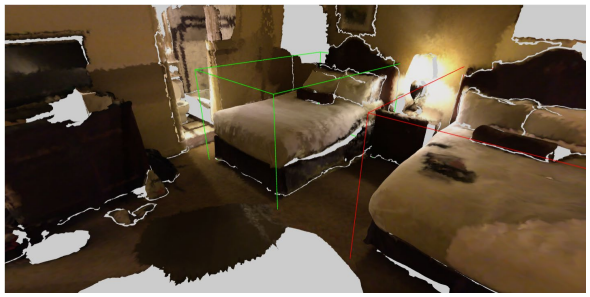
```
import numpy as np
objects = {
    '5': {'instance_label': 'ping pong table', 'color': [0.36, 0.36, 0.39], 'center': [4.87, 3.5, 0.59], 'size': [2.86, 2.68, 0.86]},
    '9': {'instance_label': 'table', 'color': [0.82, 0.71, 0.62], 'center': [4.74, 6.68, 1.1], 'size': [0.75, 0.69, 0.08]},
    '10': {'instance_label': 'table', 'color': [0.4, 0.38, 0.32], 'center': [1.35, 3.34, 0.47], 'size': [1.16, 1.2, 0.76]},
    '11': {'instance_label': 'table', 'color': [0.44, 0.4, 0.35], 'center': [2.89, 1.2, 0.72], 'size': [0.7, 0.71, 1.03]},
    '17': {'instance_label': 'picture', 'color': [0.52, 0.47, 0.39], 'center': [6.89, 3.25, 1.26], 'size': [1.81, 1.67, 1.88]},
    '18': {'instance_label': 'picture', 'color': [0.38, 0.42, 0.33], 'center': [2.49, 0.78, 1.69], 'size': [0.57, 0.55, 0.63]},
    '19': {'instance_label': 'picture', 'color': [0.31, 0.32, 0.26], 'center': [0.53, 2.83, 1.63], 'size': [0.67, 0.63, 0.53]}
}

characteristic = 'largest'
object_label = 'picture'
closest_object = 'ping pong table'

# Find the largest picture
possible_pictures = {k: v for k, v in objects.items() if v['instance_label'] == object_label}
# Determine the closest picture to the blue ping pong table (object id 5)
ping_pong_table = objects['5']
largest_picture_id = max(possible_pictures.keys(), key=lambda k: possible_pictures[k]['size'][0])
object_id = largest_picture_id

object_id
```

Figure 7. Example (b) for our method.

ing process when the utterance includes viewpoint-related information (*looking into the room from the door*), locating the circular table on the right from this viewpoint.

9. Ablation Study of 3DGS Training

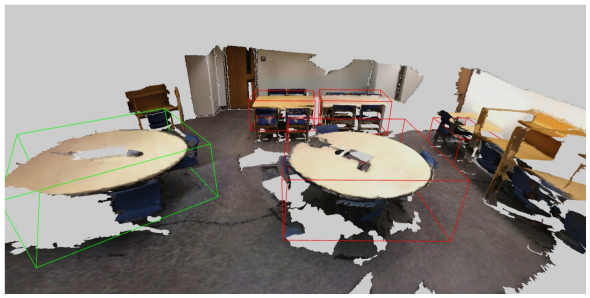
Frame Interval Analysis. We investigated the impact of training frame interval on the reconstruction of Scene_00001. With densification_interval at 500, and other 3DGS training parameters held constant, we varied the frame interval and recorded performance metrics as shown in Table 10. Training time and model size remained consistent across frame intervals, while L1 loss and PSNR varied significantly. Increasing the frame interval improved L1 loss and PSNR. However, this improvement is deceptive. Larger frame intervals reduce scene information, resulting in lower rendering quality despite the improved metrics. The improved metrics are attributed to model overfitting on sparser input views, memorizing views rather than learning a robust 3D representation. Smaller frame intervals yielded significantly better visual results, exhibiting more detail and fewer artifacts. Therefore, a smaller frame interval is crucial for high-quality reconstruction.



Utterance: It is the bed with fewer items on it. It is also closest to the bathroom.

```
import numpy as np
objects = {
    '20': {'instance_label': 'bed', 'color': [0.32, 0.26, 0.19], 'center': [3.11, 2.41, 0.8], 'size': [2.48, 2.48, 1.51]},
    '21': {'instance_label': 'bed', 'color': [0.33, 0.25, 0.19], 'center': [1.34, 3.98, 0.82], 'size': [2.47, 2.52, 1.52]},
    '35': {'instance_label': 'bathroom vanity', 'color': [0.58, 0.48, 0.36], 'center': [6.05, 1.69, 0.87], 'size': [1.23, 1.62, 0.63]}
}

def find_closest_bed(objects):
    closest_bed_id = None
    closest_distance = float('inf')
    bathroom_position = [6.05, 1.69, 0.87]  # Center of the bathroom vanity
    for obj_id, properties in objects.items():
        if properties['instance_label'] == 'bed':
            distance = ((properties['center'][0] - bathroom_position[0]) ** 2 +
                        (properties['center'][1] - bathroom_position[1]) ** 2 +
                        (properties['center'][2] - bathroom_position[2]) ** 2) ** 0.5
            if distance < closest_distance:
                closest_distance = distance
                closest_bed_id = obj_id
    return closest_bed_id

object_id = find_closest_bed(objects)
object_id
```

Figure 8. Example (c) for our method.



Utterance: if looking in the room from the door, the circular table on the right.

```
import numpy as np
objects = {
    '15': {'instance_label': 'table', 'color': [0.67, 0.52, 0.32], 'center': [4.42, 2.53, 0.56], 'size': [1.69, 1.43, 0.76]},
    '16': {'instance_label': 'table', 'color': [0.56, 0.48, 0.4], 'center': [2.88, 3.22, 0.6], 'size': [1.81, 1.57, 0.78]},
    '17': {'instance_label': 'table', 'color': [0.77, 0.72, 0.64], 'center': [4.56, 6.26, 1.09], 'size': [1.86, 1.88, 0.14]},
    '18': {'instance_label': 'table', 'color': [0.48, 0.44, 0.34], 'center': [1.35, 5.26, 0.59], 'size': [1.2, 1.41, 0.71]},
    '19': {'instance_label': 'table', 'color': [0.72, 0.64, 0.54], 'center': [7.63, 5.23, 1.19], 'size': [1.89, 1.91, 0.15]},
    '25': {'instance_label': 'door', 'color': [0.26, 0.15, 0.07], 'center': [5.11, 0.45, 1.12], 'size': [0.85, 0.42, 1.92]}
}

# Function to ground the object based on the utterance
# Extract the target object and its spatial relationship
object_id = None
for obj_id, obj in objects.items():
    if obj['instance_label'] == 'table':
        obj_position = np.array(obj['center'])
        # 'right' means in positive x-direction
        door_position = np.array(objects['25']['center'])
        if obj_position[0] > door_position[0]:
            object_id = obj_id
            break

object_id
```

Figure 9. Example (d) for our method.