

Visual Programming for Zero-shot Open-Vocabulary 3D Visual Grounding

Supplementary Material

Contents

The following two items are included in the supplementary material:

- Visualization examples for zero-shot 3DVG in Section 1.
- Prompts for generating 3D visual program in Section 2.



Query: This is a storage shelf. It is white in color and is above a desk with a chair in front of it.

Answer: Based on the description, we are looking for a storage shelf that is white in color and is above a desk with a chair in front of it.

Looking at the room information, we can see that there are several objects identified as shelves, chairs, and desks. However, the description also mentions that the shelf is above a desk, which means we need to look at the coordinates to determine the vertical positioning of these objects.

The objects identified as shelves are: <obj 24>, <obj 75>, and <obj 76>. The objects identified as chairs are: <obj 1>, <obj 36>, <obj 77>. The objects identified as desks are not explicitly mentioned, but we can consider objects like tables or nightstands as potential desks: <obj 18>, <obj 58>.

Now, we need to find a combination of these objects where the shelf is above a desk and there is a chair in front of it. This requires the shelf to have a higher z-coordinate (vertical position) than the desk, and the chair to have similar x and y coordinates (horizontal position) as the desk.

After analyzing the coordinates, none of the objects seem to fit the description perfectly. However, the closest match could be <obj 24> (shelf), <obj 36> (chair), and <obj 18> (table as desk).

Please note that this is an approximation as the description does not perfectly match any set of objects in the room.

Answer: <obj 24>

Figure 1. Visualization example (a) for zero-shot 3DVG.

1. Visualization Examples for Zero-shot 3DVG

We provide three examples to visualize the effectiveness of the proposed two zero-shot 3DVG approaches, i.e., dialog with LLM and visual programming. Concretely, the first



Query: There is a rectangular window. It is behind a desk with monitors .

Answer: First, let's identify the desk with monitors. There are several desks and monitors in the room, but the desks with monitors on them are <obj 18>, <obj 24>, and <obj 40>.

Next, we need to find a window that is behind one of these desks. The only window in the room is <obj 11>.

Looking at the coordinates, the window <obj 11> has a coordinate of (0.1, -2.8, 1.3). The desk that is in front of this window (meaning it has a smaller y-coordinate) is <obj 24> with a coordinate of (0.1, -1.5, 0.4).

So, the desk with monitors in front of the window is <obj 24>.

Answer: <obj 24>

Program:

```
BOX0=LOC(object='rectangular window')
```

```
BOX1=LOC(object='desk with monitors')
```

```
TARGET=BEHIND(targets=BOX0, anchors=BOX1)
```

Figure 2. Visualization example (b) for zero-shot 3DVG.



Query: The horizontal blinds. The blinds are above the monitor on the right .

Program:

```
BOX0=LOC(object='horizontal blinds')
```

```
BOX1=LOC(object='monitor')
```

```
BOX2=HIGHER(targets=BOX0, anchors=BOX1)
```

```
TARGET=RIGHT(targets=BOX2)
```

Figure 3. Visualization example (c) for zero-shot 3DVG.

example, i.e., Figure 1, confirms that LLMs can effectively perform zero-shot 3DVG while also delivering commendable results. The second example, as illustrated in Figure 2, shows that LLMs may encounter limitations in the tasks requiring spatial reasoning. However, this issue can be effec-

```

Your task is to parse the description to program, thus the program can be executed to
find the target object.

Here are the functions and variables you can use to parse the description to program:

# Function Definitions
def LOC(object):
    # Implement the LOC function to locate the object in the room
    # Return the objects of query text

def LEFT(targets, anchors):
    # Implement the LEFT function to find the target object to the left of the anchor
    object
    # Return the target object to the left of the anchor object
...

variable CENTER # Implement the CENTER variable to represent the center of the room

Here are some examples of the description and the corresponding program:

[0] Description: the handicapped stall
[1] Description: the blue bag in the closet
[2] Description: end table with flowers on it
...

[0] Program:
BOX0=LOC(object='handicapped stall')

[1] Program:
BOX0=LOC(object='blue bag')
BOX1=LOC(object='closet')
TARGET=CLOSEST(targets=BOX0, anchors=BOX1)

[2] Program:
BOX0=LOC(object='end table')
BOX1=LOC(object='flowers')
TARGET=CLOSEST(targets=BOX0, anchors=BOX1)
...

Here are some tips:
# Use the provided functions to implement the logic for each description
# DO NOT create new functions, variables, or constants, or modify the provided functions,
variables, or constants, find the nearest function if not exact match
# Replace `Description` with the actual description you want to parse and generate the
target object for
# Adjust the program according to the specific requirements of each description
# Wall should not be considered as an object

```

Figure 4. Prompt for generating visual programs.

tively addressed by the visual programming approach. The third example, i.e., Figure 3, further exemplifies that the visual programming approach is capable of executing multi-step reasoning, which involves initially identifying blinds that are positioned above the monitors, followed by selecting the desired one among them.

2. Prompts for Generating 3D Visual Program

As illustrated in Figure 4, the prompts for generating 3D visual programs include four components as follows:

- *Task explanation*: We first describe the 3DVG task in natural language and provide it to the LLMs.

- *Function and variable definition*: We define a set of functions and variables corresponding to the modules in the visual programming approach, such as LOC and LEFT.
- *In-context examples*: We provide contextual examples illustrating how visual programs are structured and applied to guide LLMs.
- *Best practices and tips*: We conclude with essential tips and best practices to ensure the effectiveness of the programs, highlighting the key aspects that guarantee optimal performance.

These four components collaboratively facilitate the LLM to understand the task requirement, thereby allowing it to construct effective visual programs for the 3DVG task.