From Semantically-Reasonable Plan to Physically-Valid Manipulation: A Diffusion-Based Framework for LLM-Assisted Object Rearrangement

Chenkun Zhao, Shuo Yang, Dayou Li, Ran Song, Xiaolei Li, Penkun Wei and Wei Zhang

Contents

I Overview 1

II Prompt for LLM Reasoning 1

II-A Prompt for Object and Instruction Proposal . 1

II-B Layout Generation 2

II-C Rearrangement prompt tuning 2

II-D Detection prompt tuning 2

II-E Action Score prompt tuning 2

III Charts and Algorithm 3

III-A Spatial relation 3

III-B Hard-to-Easy Prompt Tuning via VLM 4

I. Overview

This supplementary material provides the details of the prompts used for all LLM reasoning and figures in our work. Sec. II includes object proposal, layout generation, and hard-to-easy prompt tuning. Sec. III showing our figures, algorithm and some additional materials. It is worth mentioning that we also provide a video https://youtu.be/N6JZ5vxOQ6o to demonstrate the proposed method.

II. Prompt for LLM Reasoning

A. Prompt for Object and Instruction Proposal

User Input: Assuming that you are an object picker, you need to select some objects. Based on these objects, some instructions that can be executed by a robot are generated. Here are some rules you need to follow:

<rule1>: You can use random methods or your own common sense to pick some objects, and the number of objects to be selected should not be less than 6.
<rule2>: Based on these selected objects, you need to compose some instructions.
<rule3>: All objects must be selected from candidate categories. The current categories of objects include: apple, fruit platter, banana, television, keyboard, water cup, game console, mobile phones, iPad, hat, mice, tea, calendar, board game, chess, cup, plate, hamburger, hotdog, sandwich.
<rule4>: The selected objects can generate multiple instructions, and the generated instructions are related to some objects.
<rule5> Every time you generate an instruction, you need to select objects related to the instruction and objects unrelated to the instruction.
You can also refer to some sample instructions to form new instructions.

Here are some example formats:
<example1>: generated instruction: I need to eat some food, and please tidy up this desktop.
related_objects: apple, banana, plate, hamburger.
unrelated_objects: television, keyboard, mobile phone.
<example2>: generated instruction: pick the pen on the television
related_objects: pen television.
unrelated_objects: cup, phone, baseball.

Some example instructions: 1) I need to work on this table 2) normal rearrangement 3) I need to eat some food 4) pick the cup on the phone and now you should generate some commands similar to those in these examples and generate corresponding objects.
Just tell me the instruction and selected objects.

Responds:
Generated instruction : "Organize a quick snack break."
Related objects: banana, sandwich, plate
Unrelated objects: chess, television

Extra Examples:
Instruction : "It's time for a break, and please set up the game console for some relaxation."
Related objects: game console, cup, iPad.
Unrelated objects: keyboard, fruit platter, hamburger.

Instruction : "I'm getting hungry, and could you prepare the table with a fruit platter and a hamburger?"
Related objects: fruit platter, hamburger, plate, cup.
Unrelated objects: game console, iPad.

Instruction : "Let's get some work done, and please clear the desk and organize the iPad and the keyboard for productivity."

Related objects: iPad, keyboard, cup.
Unrelated objects: game console, fruit platter, hamburger.

B. Layout Generation

User Input: Now the provided instruction is "Organize a quick snack break". The objects in the tabletop are banana, sandwich, plate, chess, television. Use your common sense on the table rearrangement. Objects unrelated to instructions can be placed directly away from the center and arranged neatly. Please tell me how to rearrangement them. You can place the relevant objects you think neatly together, and the unrelated objects can be arranged in a line on the side. You should generate a list directly and briefly.

Responds:
1. Place the banana and the sandwich on the plate, positioning it at the center of the table for easy access.
2. Arrange the chess set to one side of the table, neatly but away from the central focus to indicate that it's not part of the snack break.
3. Position the television in a way that it can be viewed from the snack area but doesn't occupy central space.

User Input: Then you will be given the 3D bounding boxes of all objects.
The bounding box format:
[xmin,ymin,zmin,xmax,ymax,zmax]
sg
<rule1>: The bounding boxes of objects should not overlap.
<rule2>: Arrange according to the requirements of the previous output.
<rule3>: The size of the bounding box after transformation should not change. For each object, it should be xmax−xmin, ymax−ymin, zmax−zmin without any change.

<output format>:
object1: [xmin,ymin,zmin, xmax,ymax,zmax]
object2: [xmin,ymin,zmin, xmax,ymax,zmax]
object3: [xmin,ymin,zmin, xmax,ymax,zmax]
...
now the input bounding boxes are:
banana [0.2,0.2,0.4,0.29,0.37,0.43],
sandwich [0.3,0.4,0.4,0.38,0.53,0.46],
plate [0.5,0.5,0.4,0.7,0.7,0.42],
chess [0.6,0.2,0.4,0.8.,0.4,0.6],
television [0.1,0.3,0.4,0.6,0.8,0.7]

Responds:
Plate [0.3, 0.3, 0.4, 0.5, 0.5, 0.42]
Banana [0.31, 0.33, 0.4, 0.4, 0.5, 0.43]
Sandwich [0.41, 0.33, 0.4, 0.49, 0.46, 0.46]
Chess [0.05, 0.1, 0.4, 0.25, 0.3, 0.6]
Television [0.7, 0.2, 0.4, 1.2, 0.7, 0.7]

C. Rearrangement prompt tuning

User Input: Scene objects: [mobile phone, chess set, notebook, pen, plate, chocolate]
I will give you an instruction, ask you to analyze it, find out the relevant objects and their relative positions in the target state (remember you should include all the objects in the scene), and summarize the scene described by this instruction in no more than 3 words.
The output format should be: [[manipulation1:object1 ,...],[manipulation2:object1 ,...],...,[summarization of the scene]], where "manipulation" means the operation that needs to be performed on the object.

Now the instruction is: I just finished work. I'm a little tired and need some entertainment.
Please help me bring my mobile phone and chess set, take away notebooks and other irrelevant things, and bring some chocolates.

Responds:
[[bring:mobile phone, chess set, chocolate],
[leave: plate], [take away:notebook, pen],
[relaxation setup]]

D. Detection prompt tuning

User Input: This picture contains some cards, and now I need you to help me find which one has RSS written on it. Then you need to tell me about its features, including useful information such as color, shape, position, size, etc. You need to reply to me according to the template below:

<output format>:
name:
color:
size:
shape:
position:

Responds:
name: RSS
color: Red
size: Medium compared to other cards in the image
shape: Square
position: Towards the bottom center, between the red circle (CVPR) on the left and the blue circle (ICRA) on the right.

E. Action Score prompt tuning

User Input: You are now an object operator, and you need to figure out the order
in which to operate the objects based on your understanding of the scene.

```
I will provide you with the initial and final positions
of each object,
and based on your knowledge, you can infer the order in
which to operate these objects.
And attach the corresponding ranking score between 0-1

Placed_objects means these objects have been placed at
the goal location
Placed_objects =[]
Remaining_objects means these objects are not placed at
the goal position
Remaining_objects=[plate, apple]

Scene description
object ((cur_pos, goal_pos))
plate ((0.1,0.2, 0.4), (0.4,0.2,0.4) )
apple ((0.4,0.2,0.4) , (0.1,0.2, 0.4))

output format
You need to operate the objects as the following
sequence
[(object2, score2), (object1, score1), ......]

Responds:
[(apple, 0.7), (plate, 1.0)]
```

```
User Input: You are now an object operator, and you
need to figure out the order
in which to operate the objects based on your
understanding of the scene.
I will provide you with the initial and final positions
of each object,
and based on your knowledge, you can infer the order in
which to operate these objects.
And attach the corresponding ranking score between 0-1

Placed_objects means these objects have been placed at
the goal location
Placed_objects =[]
Remaining_objects means these objects are not placed at
the goal position
Remaining_objects=[plate, apple]

Scene description
object ((cur_pos, goal_pos))
plate ((0.3,0.5, 0.4), (0.4,0.2,0.4) )
apple ((0.4,0.2,0.4) , (0.1,0.2, 0.4))

output format
You need to operate the objects as the following
sequence
[(object2, score2), (object1, score1), ......]


Responds:
[(apple, 0.7), (plate, 1.0)]
```

```
User Input: You are now an object operator, and you
need to figure out the order
in which to operate the objects based on your
understanding of the scene.
I will provide you with the initial and final positions
of each object,
and based on your knowledge, you can infer the order in
which to operate these objects.
And attach the corresponding ranking score between 0-1

Placed_objects means these objects have been placed at
the goal location
Placed_objects =[apple]
Remaining_objects means these objects are not placed at
the goal position
Remaining_objects=[plate]

Scene description
object ((cur_pos, goal_pos))
plate ((0.3,0.5, 0.4), (0.4,0.2,0.4) )
apple ((0.1,0.2, 0.4), (0.1,0.2, 0.4))

output format
You need to operate the objects as the following
sequence
[(object2, score2), (object1, score1), ......]

Responds:
[(plate, 1.0)]
```

III. Charts and Algorithm

A. Spatial relation

The object can be defined as $O = (l_x, l_y, l_z, \theta)$ using *cuboid representation* where (l_x, l_y, l_z) is the size of the object along the x,y,z axis respectively. θ is the rotation around z axis. So we can get the lowest and highest point by object orientation.

Fig. 1. Examples of the object library. We select more than 400 household objects from the BEHAVIOR-1K dataset to construct the object library for data generation.

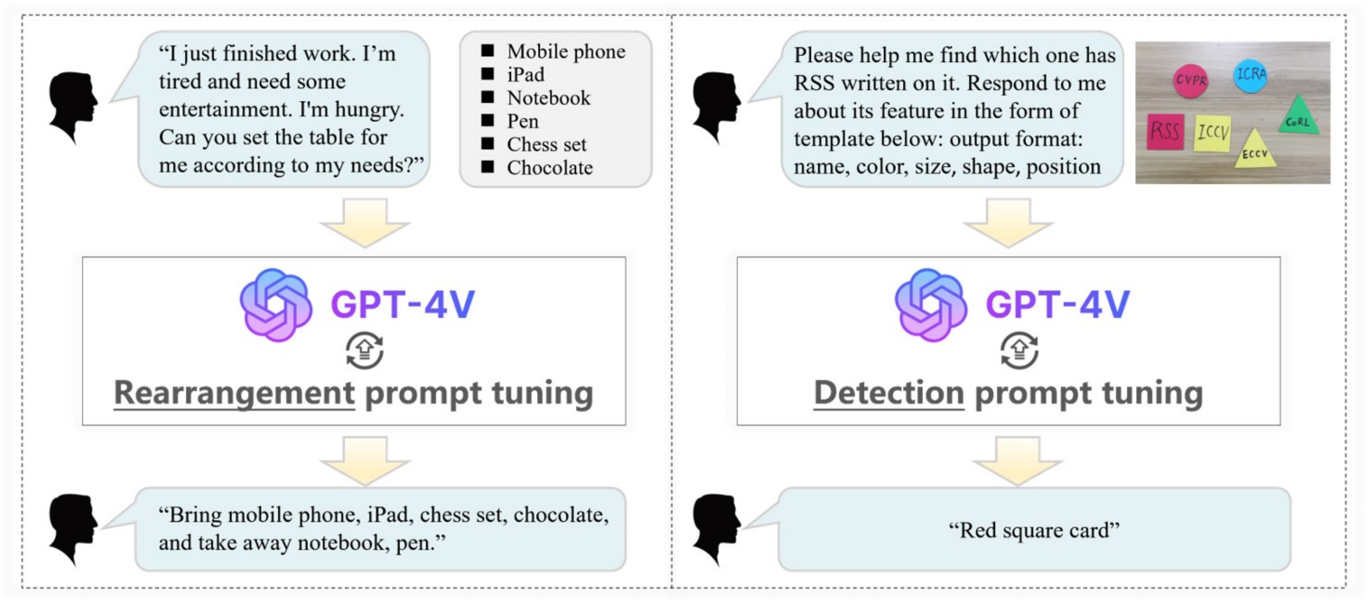
"I just finished work. I'm tired and need some entertainment. I'm hungry. Can you set the table for me according to my needs?"
Mobile phone
iPad
Notebook
Pen
Chess set
Chocolate
GPT-4V
Rearrangement prompt tuning
"Bring mobile phone, iPad, chess set, chocolate, and take away notebook, pen."
Please help me find which one has RSS written on it. Respond to me about its feature in the form of template below: output format: name, color, size, shape, position
CVPR
ICRA
RSS
ICCV
CoRL
ECCV
GPT-4V
Detection prompt tuning
"Red square card"

Fig. 2. Illustration of the hard-to-easy prompt tuning strategy. We build this prompt tuning strategy upon GPT-4V where the core idea is to reinterpret a hard prompt into an easy-to-understand one by leveraging the commonsense and visual reasoning capability of GPT-4V.

$$\begin{aligned} p &= R_\theta[-\frac{l_x}{2},-\frac{l_y}{2},-\frac{l_z}{2}]^T + [x+\frac{l_x}{2},y+\frac{l_y}{2},z+\frac{l_z}{2}]^T, \\ q &= R_\theta[\frac{l_x}{2},\frac{l_y}{2},\frac{l_z}{2}]^T + [x+\frac{l_x}{2},y+\frac{l_y}{2},z+\frac{l_z}{2}]^T \end{aligned} \tag{1}$$

R_θ is the rotation around the z axis. An object is represented by (p,q,θ). Suppose there are two objects A and B represented as $O_a=(p_1,q_1,\theta_1)$ and $O_b=(p_2,q_2,\theta_2)$. We simply introduce 4 spatial useful relations:

$$on: z_{p1} = z_{s2} \wedge \frac{p_1+q_1}{2} \in_{xy} O_2; \tag{2}$$
$$above: z_{q2}+d_{min-above} \leq z_{p1} \leq z_{q2}+d_{max-above} \tag{3}$$
$$\wedge \frac{p_1+q_1}{2} \in_{xy} O_2; \tag{4}$$
$$under: z_{s1} < z_{s2} \wedge O_1 \cap_{xy} O_2 \neq \emptyset; \tag{5}$$
$$contain: z_{p1} < z_{p1} < z_{q2} \tag{6}$$

where $d_{min-above}$, $d_{max-above}$ are distance thresholds, $p \in_{xy} C$ represents point p is inside object C on the x-y plane. In our settings, if A is currently on or included by object B, A will get a higher score to encourage priority movement. For the goal position, if A is above B, A will be blocked.

B. Hard-to-Easy Prompt Tuning via VLM

The proposed diffusion model is trained with the automatically generated data. During inference, we input the initial configuration and a specific instruction into the diffusion model to predict the goal state. In the simulated environment, we can easily obtain the attributes of objects through API interface. While in real-world scenarios, we employ OWLv2 as object detector to acquire object attributes. Although such a strategy is widely used in recent methods for object rearrangement, it did not perform very well in real-world rearrangement tasks due to the limited reasoning capability of object detectors. In practice, open-vocabulary object detectors can only understand relatively simple query words (e.g., '*apple*', '*cup*', '*knife*') and tend to perform worse if a complex prompt is given (e.g., '*select the object that might be the most dangerous.*'). It is noteworthy that existing instruction-guided rearrangement methods cannot understand complex prompts well.

VLMs have shown powerful reasoning capability. Thus we propose to use GPT-4V for enhancing the rearrangement capability of our method across diverse instructions and objects. As illustrated in supplementary material Fig. 2, we design a hard-to-easy prompt tuning strategy based on GPT-4V. Its core idea is to reinterpret a hard prompt into an easy-to-understand one via GPT-4V. The prompt tuning strategy contains two components. First, we perform language reasoning via GPT-4V to tune the input prompt of our diffusion model. Given the raw prompt I and object attributes o_i (including category, position, rotation, and size, on demand available), we construct a query prompt in accordance with a hand-designed template and then forward it into GPT-4V, which finally outputs a tuned prompt. Second, we perform visual reasoning via GPT-4V to tune the input prompt of the open-vocabulary object detector OWLv2. In an inference case, if OWLv2 outputs the prediction results with low confidence values, the input prompt will be refined by GPT-4V. As shown in the right half of supplementary material Fig. 2, we input the scene image and a designed prompt into GPT-4V which reinterprets a complex prompt into a simple one represented as four object attributes (size, shape, position, and color). Please refer to the supplementary material for the details of the prompt engineering.

Algorithm 1 From Semantically-Reasonable Plan to Physically-Valid Manipulation

```
 1: Input: Language goal description $L$, and initial messy scene $I$;
 2: Detection Prompt tuning $o_{raw} \rightarrow o_{tune}$;
 3: Rearrangement Prompt tuning $L_{raw} \rightarrow L_{tune}$;
 4: Generate object centric goal via diffusion model
    $\{o_1, o_2, ..., o_N\} \leftarrow f_\theta(\widetilde{X}, L_{tune})$;
 5: while $len(Remaining\_objects) > 0$ or $task\ not\ failure$ do
 6:     Generate action candidate score via LLM;
        LLM$(S, P, R \| \phi_p) \rightarrow \{score_i \in (0,1)\}$;
 7:     Refine it via physical heuristics;
 8:     Select $best\_score$ action candidate;
        $(object_i, cur\_pos, goal\_pos) \leftarrow argmax(\{score_i\})$
 9:     $\tau \leftarrow$ trajectory diffusion $\pi_f$
10:     Upadate $scene\ description$
11:     $Remaining\_objects$.pop($object_i$)
12: end while
13: Execute generated trajectory sequence $\{\tau_1, \tau_2, ..., \tau_n\}$;
```