

This appendix provides additional details and supplementary materials for **TP-MDDN: Task-Preferred Multi-Demand-Driven Navigation with Autonomous Decision-Making**, supporting the main content of the paper. The section is organized as follows: Appendix 1 presents the prompt engineering details of the large models.

1 Prompt Engineering Details

Figure 2 is used to guide the large language model in decomposing complex instructions into exactly three subtasks. The format enforces structured output for consistency in downstream processing. Figure 3 is used to guide the large language model in selecting the next subtask and associated object-position pair for the Visual Language Navigation (VLN) agent. Figure 1 shows a prompt that guides the model to identify the most recently completed subtask and output completion statuses for all subtasks in JSON format based on visual and contextual input.

You are an expert AI assistant evaluating the completion status of a sequence of subtasks performed by a robot agent in a simulated environment.

Your goal is to:

1. Identify WHICH subtask from the plan was most likely JUST attempted or completed based on the CURRENT image and context.
2. Provide reasoning for this identification.
3. Determine the final status (True/False) for EACH subtask in the original plan based on the final image and context.

Context includes:

1. The overall high-level instruction.
2. The planned sequence of subtasks.
3. Previous task history (contextual).
4. The CURRENT image view.

Analysis:

- Analyze the CURRENT image.
- Compare the image and history against the subtask plan.
- Identify the most recent action evident in the image or implied by the state leading to "Done".
- Evaluate the final completion status (true/false) for ALL original subtasks, considering the net result shown in the final image and history.

Output your response ONLY as a single, valid JSON object with the following top-level keys:

- "last_evaluated_subtask": (String) The exact name/description of the subtask most likely just attempted/completed.
- "last_evaluation_reasoning": (String) Your reasoning for identifying the 'last_evaluated_subtask' and its implicit status.
- "all_subtask_statuses": (Object) A JSON object where:
 - Keys MUST be the exact string descriptions of ALL subtasks from the original "Subtask Plan".
 - Values MUST be JSON standard lowercase boolean literals: 'true' or 'false', representing the final status of each task.

Example CORRECT Output Format:

```
'''json
{
  "last_evaluated_subtask": "Put apple on sofa",
  "last_evaluation_reasoning": "The image shows the apple resting on the sofa cushion near the agent, which directly corresponds to the final subtask.",
  "all_subtask_statuses": {
    "Navigate to the kitchen table": true,
    "Find the apple on the table": true,
    "Pick up the apple": true,
    "Navigate to the living room sofa": true,
    "Put apple on sofa": true
  }
}
'''
```

Figure 1: Prompt template for subtask completion tracking.

Your task is to decompose complex instructions into exactly three component parts. Please follow this output format strictly: {1: subtask1, 2: subtask2, 3: subtask3}. Do not include any extra text, explanations, or symbols (such as emojis). Output only the dictionary with three numbered subtasks as strings.

Example Input Instruction: "I need a way to create a cozy atmosphere, stay organized, and add some artistic touch."
Example Output: {1: "Create a cozy atmosphere", 2: "Stay organized", 3: "Add an artistic touch"}

Figure 2: Prompt template for instruction decomposition.

You are an intelligent assistant for a Visual Language Navigation (VLN) agent executing long-term tasks in an environment. Your goal is to determine the next immediate action based on the overall plan and current situation.

Context:

Overall Goal (User Request):

{user_request}

Subtask Plan (Predefined Breakdown):

{formatted_subtasks}

Current Subtask Status:

(Indicates whether each subtask in the plan has been completed)

{formatted_statuses}

Objects Encountered So Far:

(Objects detected in the environment with their 2D positions, with the confidence level of the object in parentheses after the object noun)

{objects_encountered}. {extra_instruction}.

Agent's Movement History:

(The relevant objects and target positions from previous steps, and the execution result of each action.

Format: Relevant Object(s): [list of object names], Target Position(s): [(x, y) coordinates], Execution Result: [success or failure description]

{formatted_history}

Your Task:

Based on the provided context, determine the **next subtask** to execute. Identify the **relevant object(s)** needed for this subtask from Objects Encountered So Far and their **target 2D position(s)**.

Instructions & Constraints:

1. **Identify the Next Subtask:** Find the **first** subtask in the 'Subtask Plan' list that has a status of 'false' in the 'Current Subtask Status'. This is the 'Next Subtask'.
2. **Do Not Repeat Completed Tasks:** Ensure the selected 'Next Subtask' has a status of 'false'.
3. **Identify Relevant Objects:** Select objects from the analysis of 'Objects Encountered So Far' that may meet the requirements of the 'Next Subtask'. The output object must be one of the objects encountered so far.
 - Check the agent history and do not output the same object as before.
 - Observing the agent history, if the previous object failed to execute, it can continue to execute. If it has already failed twice, temporarily abandon the object and explore other options first.
 - Do not output objects that are in subtasks but not in 'objects surrounded so far', such as food.
4. **Find Target Positions:** Look up the 2D positions of these 'Relevant Object(s)' in the 'Objects Encountered So Far' list.
 - Do not repeatedly output a position.
 - Only one position can be output. If multiple positions are available for an object, prioritize positions that have not been visited in the 'Agent's Movement History' and that are likely to lead to successful completion of the subtask.
5. **Avoid outputting objects and locations that are the same as those in the agent history**
6. **Output Format:** Provide the output **strictly** in the following format:

Tips:

1. You can explore places such as door, cabinet, bed, sofa, etc
2. It is best not to check objects with unclear descriptions such as bin, basin, stool, cube, etc.
3. Do not output objects that are in subtasks but not in 'objects surrounded so far', such as food.
4. {extra_instruction}

Required Output Format:

- **Next Subtask:** [Name of the identified next subtask]
- **Relevant Object(s):** [List of relevant object names, e.g., ["apple", "table"], you must choose one from the 'Objects Encountered So Far']
- **Target Position(s):** [List of corresponding (x, y) coordinates from Objects Encountered, e.g., [(1.5, 2.3), (4.0, 5.1)], you need to output the 2D position of the selected object]

Figure 3: Prompt template for target object decision.