Implementation Details

1 Look Forward Module

The *look forward* module serves as a global parser for the navigation instruction. It takes the entire instruction as input and decomposes it into a sequence of atomic actions, ensuring that all actions and landmark descriptions are captured. Each sub-instruction produced by the module must contain at least one action (such as "go through the door" or "turn left"), and, when mentioned, it must also include specific landmarks that ground the action in the environment. This global view prevents the omission of actions and ensures the integrity of the instruction decomposition.

As illustrated in Figure 1, the system first provides an internal *system prompt* describing how to decompose the input instruction: sub-instructions should be complete sentences, each covering a single action and any referenced landmarks. The *user prompt* supplies a navigation instruction, for example: "Go through the door and turn left. Go to the left of the stairs. Stop in the doorway to the left of the white double doors." The module then outputs a list of sub-instructions paired with their corresponding landmarks. These sub-instructions ensure that each action is explicit and that any environmental details are retained. A typical output contains approximately three to eight sub-instructions, depending on the complexity of the original instruction, providing a structured and human-readable sequence for downstream processing.

Step 1: Look Forward (Global View)

System Prompt: You are an action decomposition expert. Your task is to decompose the whole instruction into a series of sub-instructions and all actions in the given navigation instruction. You need to ensure the integrity of each action. You need to make sure the sub-instructions are complete, and include the details of the current environment if it is mentioned in the instruction. Your answer must consist ONLY of a series of labeled action phrases without begin sentence. For each sub-instruction, it should involve at least one action, and all the description of the environment related to the same location. A typical answer should involve 3 to 8 sub-instructions.

User Prompt: Can you decompose actions in the instruction "*Go through the door and turn left. Go to the left of the stairs. Stop in the doorway to the left of the white double doors.* "?

Decomposed Sub-Instructions with Corresponding Landmarks
1. Go through the door. | ['the door']
2. Turn left. | ['There are no landmarks in the instruction "2. Turn left."']
3. Go to the left of the stairs. | ['the stairs']
4. Stop in the doorway to the left of the white double doors. | ['doorway to the left of the white double doors', 'white double doors']

Figure 1: We designed the sub-instruction separation prompts for the *look forward* module.

2 Look Now Module

The *look now* module provides a fine-grained, step-wise navigation process conditioned on the current visual observation. At each step, the system integrates the parsed sub-instruction from the *look forward* stage with the accumulated navigation history and the scene observations around the agent. It constructs a detailed system prompt that enumerates the visible candidate viewpoints, their semantic and geometric properties, and the agent's current spatial context. The multimodal large language model (MLLM) is then prompted to reason about which direction best aligns with the current sub-instruction, producing a structured output that contains: (1) a chain-of-thought justification ("Thought"), (2) the estimated distance toward the predicted direction, (3) the predicted next viewpoint, and (4) a binary completion estimation indicating whether the current sub-instruction is complete.

As illustrated in Figure 2, the MLLM compares candidate viewpoints by analyzing spatial cues (e.g., door positions, hallway direction) and aligns them with linguistic landmarks in the instruction. In this example, the model determines that the next action "turn left" should follow after moving through the door. It selects the viewpoint corresponding to the open hallway (Direction 1) as the most consistent continuation and estimates that the local goal is reached.

Step 2: Look Now (Local View)

System Prompt Content

[Task Background]

You are an embodied robot that navigates in the real world. You need to explore between some places marked with IDs and ultimately find the destination to stop.

[Input Definitions]

I will give you one instruction and tell you landmarks. I will also give you navigation history for reference. You can observe current environment by scene descriptions, scene objects and possible existing landmarks in different directions around you. Each direction contains direction viewpoint ids you can move to. Your task is to predict moving to which direction viewpoint. Each direction viewpoint has an image that you can see.

[Output Requirements]

If you can already see the destination, estimate the distance between you and it. If the distance is far, continue moving and try to stop within 1 meter of the destination. Your answer includes four parts: "Thought", "Distance", "Prediction" and "Completion Estimation". In the "Thought", you should think as detailed as possible following procedures: ... Then, please make decision on the next viewpoint in the "Prediction". Your decision is very important, must make it very carefully. You need to double check the output in "Prediction:". The output must be in the Candidate Viewpoint IDs without any other words. You also need to double check the output in "Thought". The output must be a single paragraph. After finished all the above steps, you need to estimate the completion of the instruction based on the 'Instruction', 'Next instruction', 'Landmarks', your past 'Navigation History', 'Current Environment', and the provided images. Please think carefully about the 'Distance' when you estimate the completion of the instruction. If your current distance to the destination is very far, you should answer 'No'. If your current distance to the destination is close and you think you are ready to walk towards the landmarks of next instruction, you should answer 'Yes'."

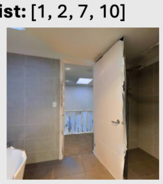
User Prompt Content

Instruction: 1. Go through the door. **Landmarks**: ['the door'] **Next instruction:** 2. Turn left.

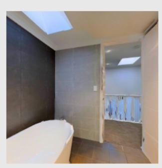
Navigation History: Step 1 Observation: Direction Font Left, range(left 15 to left 45) Modern white bathroom: from nearest to farthest—toilet (0.8 m), bathtub (1.2 m), door (2.5 m), window (3.5 m), shower (4.5 m); also includes a sink, floor, and a wall lamp. Thought: Moved straight from the bathroom toward and through the open doorway, exiting into the brighter hallway with a railing, taking the most direct path and bypassing the tub and side wall.

Current Environment: ['Direction 1 Direction Viewpoint ID' 1 in Step ID 2: Elevation: Eye Level, Scene Description: The image contains a door, a shower, a wall, and a ceiling. The door is in the foreground, the shower is to the right, and the wall and ceiling are in the background. The door is approximately 1 meter from the camera, the shower is about 2 meters from the camera, and the wall and ceiling are further away, likely around 5 meters. Scene Objects: bath | bathroom | bathroom accessory | doorway | floor | glass door | lead to | restroom | room | screen door | shower | sink | slide | tile wall | tub | walk; ', ...]

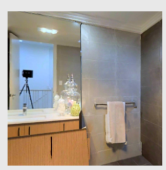
Candidate Viewpoint IDs List: [1, 2, 7, 10]



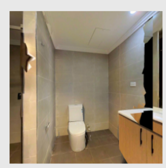
VP1



VP2



VP7



VP10

MLLM Response Content

Thought: The instruction is to go through the door, and the next step will be to turn left in the hall. Comparing directions, 1 and 2 both show the open doorway leading to the bright hall with a railing, but Direction 1 places the door closest (about 1 m) and directly ahead, while Directions 7 and 10 look further into the bathroom. To follow the plan from the history and exit toward the hall efficiently, I should move toward Direction 1 to pass through the door.

Distance: ~1 m to the doorway/hall. **Prediction:** 1 **Completion Estimation:** Yes

Figure 2: Task prompts for the local-level *look now* module.

3 Look Backward Module

When the progress estimation result from *look now* module shows 'yes', ThreeStep Nav will start the *look backward* module. As illustrated in Figure 3, the *look backward* module acts as a global auditing mechanism that evaluates whether the executed actions have satisfied the current sub-instruction and determines how to transition to the next one. While the *look forward* and *look now* modules operate on linguistic parsing and local visual reasoning, respectively, the *look backward* stage explicitly integrates code-level semantics and navigation control logic. The model is prompted with both the high-level navigation history and the code implementation snippets that define meta-navigation behaviors, such as *Continue*, *Stay*, *Look-Around*, and *Backtrack*. By understanding the effects and intended purposes of these functions, the system can reason about which control operation best fits the current context.

Step 3: Look Backward (Global View)

System Prompt Content

You are a code-aware navigation decision agent that understands implementation details of four different meta navigation abilities.

User Prompt Content

[Navigation abilities]

CONTINUE Implementation:

```
@staticmethod
def stay_with_current_instruction(context: Dict[str, Any]) -> Dict[str, Any]:
    """
    Continue working on the current sub-instruction.
    Implementation:
    1. Maintain current_action_idx
    2. Preserve navigation history
    3. Continue accumulating visual evidence
    4. Update confidence metrics
    Returns:
    Context with updated confidence but same instruction
    """
    context['attempts_on_current'] = context.get('attempts_on_current', 0) + 1
    context['state'] = 'navigating'
    return context
```

Capability: Continue to Next Instruction.
Purpose: Progress to the next sub-instruction when current is completed.
Effects: Increments action index, Resets navigation history, Clears visual memory, Starts fresh for new sub-goal
When to use: When current landmarks have been found and instruction is satisfied.

BACKTRACK Implementation:

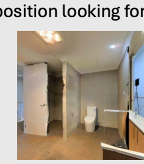
```
@staticmethod
def backtrack_to_previous(context: Dict[str, Any], env_actions_history: List) -> Dict[str, Any]:
    import math
    if len(env_actions_history) > 0:
    last_action = env_actions_history.pop()
    # Create reverse action
    reverse_action = {
        'action': {
            'action': 4, # Move action
            'action_args': {
                'angle': (last_action['action']['action_args']['angle'] + math.pi) % (2 * math.pi) - math.pi,
                'distance': last_action['action']['action_args']['distance']
    }}}
    # Update context
    if len(context.get('nav_history', [])) > 0:
        context['nav_history'].pop()
    if len(context.get('chosen_images', [])) > 0:
        context['chosen_images'].pop()
    context['backtrack_action'] = reverse_action
    context['state'] = 'backtracking'
    return context
```

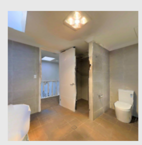
Capability: Backtrack to Previous Position
Purpose: Undo last movement and return to previous position
Effects: ...
When to use: ...

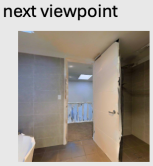
STAY Implementation: ...
LOOK-AROUND Implementation: ...

[Input Definitions]

Given this understanding of what each action actually does in the code and considering the current context:
Finished Instructions: 1. Go through the door.
History: Step 1 Observation: ... -> Step 2 Observation: ...
Image sequence descriptions:
Image 0: Initial position: Agent standing at start point looking forward
Image 1: Step 1: Agent at previous position looking front-left (30°) towards chosen next viewpoint
Image 2: Step 2: Agent at previous position looking forward towards chosen next viewpoint


Start point


Node 1


Node 2

[Output Requirements]

What is the most appropriate navigation decision? Consider the actual code effects, not just the conceptual purpose. Match your situation to the examples above.
Format your response as:
Reasoning: [Your detailed reasoning]
Confidence: [0-10]
Decision: [Continue/Stay/Backtrack/Look Around]

MLLM Response Content

Reasoning: The sub-instruction is "Go through the door." The door landmark has been located and traversed—history notes passing through the open bathroom door into the brighter hallway with a railing, and the latest view indicates the doorway now leading back into the bathroom. Since the current instruction is satisfied, the correct coded action is CONTINUE to advance to the next sub-instruction, accepting its effects (increment index and reset history/memory).
Confidence: 9
Decision: Continue

Figure 3: An example of the complete prompting process in the *look backward* module. The system integrates both visual and code-level contexts to decide how to proceed after executing a sub-instruction. The prompt includes code definitions for four meta-navigation abilities—*Continue*, *Stay*, *Look-Around*, and *Backtrack*—together with sequential observations of the agent's trajectory. Here, the MLLM reasons that the sub-instruction "go through the door" has been fulfilled and correctly selects *Continue*, indicating readiness to progress to the next navigation goal.