EmbRACE-3K: Embodied Reasoning and Action in Complex Environments (Supplementary Material)

1 More Details on Data Collection and Benchmark Construction

1.1 Task Instruction Generation

Figure 3 shows the prompt template used to generate natural language navigation tasks with Gemini. The prompt encodes task types, visibility constraints, and spatial-semantic reasoning requirements, guiding the generation of grounded instructions aligned with the agent's egocentric observation. Each task is conditioned on a sampled agent pose, surrounding object metadata, and a list of reasoning constraints, ensuring task diversity and consistency.

Standard navigation tasks. For Type 0–3 tasks, we provide Gemini-2.5-Pro [2] with the current egocentric image and a list of candidate objects within a 1000-meter radius. Based on this input, Gemini generates instructions falling into four predefined categories:

- **Type 0 (Basic):** The target is clearly visible and reachable without exploration.
- **Type 1 (Exploration):** The target is not initially visible and must be discovered through search.
- **Type 2 (Dynamic Spatial Semantic):** The target is visible but referred to via relative or ordinal spatial language (e.g., "the second chair," "the one on the right").
- **Type 3 (Multi-stage):** Two visible targets must be remembered across steps and approached in sequence.

Interaction-based tasks. For Type 4 tasks involving object interaction, we follow a hybrid generation process that combines manual layout curation with template-guided instruction creation. Interaction tasks are divided into:

- **Type 4.1 (Open Door):** We manually place the agent near an interactive door. If the door is clearly visible from the agent's initial view, a visible-door template is selected. Representative templates include:

  "Walk to the door ahead and open it.";
  "Step forward and open the door visible in the scene.";
  "Move toward the obvious door in front of you and open it.";
  "You're facing a door. Approach and open it."

  When the door is not visible initially, we sample from an exploratory search template set designed to encourage navigation and search behavior:

  "Look around and locate a door to open.";
  "Search the area and open a nearby door once you find it.";
  "Explore the environment and open the first door you find.";
  "Walk around until a door is in view, then go open it."

- **Type 4.2 (Pick & Drop):** Graspable objects (e.g., trash bags, candle holders, burlap sacks) are placed near the agent, and distinct landmarks are selected as drop targets. Instructions emphasize both object manipulation and relative spatial reasoning. Representative templates include:

  "Pick up the trash bag and place it near the red car.";
  "Bring the candle holder over and nestle it into the far-right stone corner.";
  "Carry the burlap sack and drop it next to the table on the far-left side closest to you."

Final instruction curation. All generated instructions are manually reviewed for scene consistency, target uniqueness, and spatial clarity. This ensures that each command remains executable and unambiguous given the visual field of the agent. The final instruction set used for training and evaluation supports realistic yet controlled embodied tasks, with clear grounding in egocentric perception and task semantics.

1.2 Step-wise Reasoning Annotation

To provide interpretable supervision aligned with agent behavior, we annotate each step of the demonstration trajectory with natural language reasoning that explains the agent's decision at that moment. These *thinking* annotations simulate the internal deliberation process of an embodied agent, grounded in egocentric perception and constrained by task semantics.

As illustrated in Figure 4, we construct a prompt that includes the task instruction, the agent's complete egocentric image sequence, and the full history of executed actions. This prompt is provided to Gemini-2.5-Pro, which returns a sentence describing why the current action is appropriate given what the agent perceives and remembers. Importantly, the model is explicitly instructed to avoid using any future information and to write in the first person, as if it were the agent itself.

To ensure geometric consistency and spatial reasoning fidelity, the prompt imposes constraints on spatial language: object locations must be described using viewpoint-aligned phrases such as "on the left-hand side" or "in the center," and references like "to the left" or "right beside me" are disallowed. For multi-stage instructions, the reasoning must focus exclusively on the first subgoal until the designated `MidwayTarget` action appears in the trajectory.

This step-wise annotation process results in a rich set of perception-action-thought triplets that reflect how embodied agents might reason in closed-loop settings. These annotations serve as fine-grained supervision for training and offer a transparent lens for evaluating spatial-semantic alignment during embodied decision-making.

1.3 Data Prompts

We construct three types of data prompts for model training: (i) supervised fine-tuning (SFT) with reasoning, (ii) SFT without reasoning, and (iii) reinforcement learning (RL). All prompts simulate the embodied decision-making process based on task objectives, egocentric visual observations, and action histories.

SFT with reasoning. For step-wise reasoning supervision, we provide the model with the full task instruction, a time-ordered sequence of first-person egocentric views up to the current step, and the full trajectory of past actions. The model is expected to produce both a natural language explanation (`<think>`) and the corresponding action (`<action>`) at each step. This setup encourages learning spatially grounded, interpretable decision policies aligned with visual context and task semantics.

SFT without reasoning. For instruction-following SFT without reasoning, the prompt retains the same structure but omits the reasoning component. The model directly predicts the next action given the full observation-action history and task instruction. This setting isolates the effect of action-only learning, enabling ablation comparisons with reasoning-augmented variants.

RL training. During reinforcement learning, the same prompt format as the reasoning-free SFT setup is used. The model interacts with the environment under a policy trained via reward feedback and generates the next action at each step without explicit language supervision. The reward signal supervises the selected action and its format, encouraging alignment with the target behavior defined by the dataset.

Figures 5 and 6 illustrate examples of the two prompt formats used in SFT and RL training.

2 More Visual Details of EmbRACE-3K

To better illustrate the structure and coverage of EmbRACE-3K, we provide visualizations of representative human-annotated trajectories across all six task types. Each example includes the full

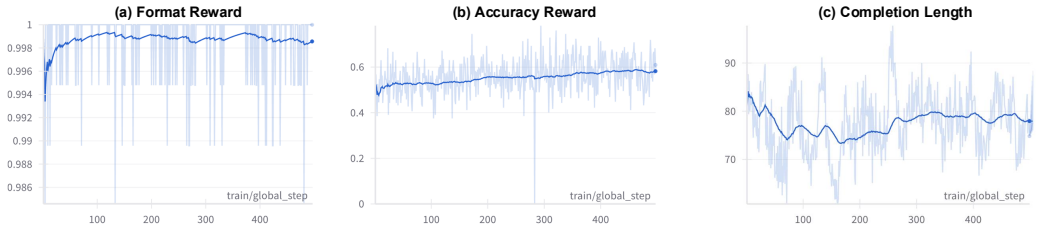

Figure 2: Training curves of three key metrics during reinforcement learning with GRPO. "Completion Length" indicates the output length of the policy model during rollout process. The policy model is initialized from the SFT checkpoint.

instruction, a sequence of egocentric first-person views, the corresponding human actions, and (where available) step-wise reasoning annotations. These demonstrations reflect the diversity of embodied behaviors and instruction formats captured in the dataset.

Figures 7 through 12 showcase how different task types correspond to distinct reasoning challenges. In particular, EmbRACE-3K is designed to probe three core limitations in current video-trained vision-language models: (i) short-sighted exploration, where agents lack persistence in visually-guided search; (ii) dynamic spatial-semantic drift, where spatial references change meaning under egocentric motion; and (iii) target forgetting, where objects that briefly leave the visual field are not recalled later in the task. The examples emphasize how task instructions, visual cues, and behavior sequences interact to reveal these difficulties.

Rather than prescribing a unique solution, the visualizations present one plausible human trajectory per task. These examples serve to concretely illustrate the design intent and diagnostic scope of EmbRACE-3K, providing interpretable reference patterns for supervised or policy-based training.

3 More Details of Experiments

3.1 SFT Training Curve

Figure 1 presents the SFT loss curve of Qwen2.5-VL-7B [1] on the EmbRACE-3K dataset. The training loss decreases steadily as the model is optimized to align egocentric observations with instruction-conditioned action decisions. The smoothed trajectory demonstrates stable convergence across training iterations, indicating the learnability of our structured prompts and the coherence of human-annotated trajectories and reasoning signals.

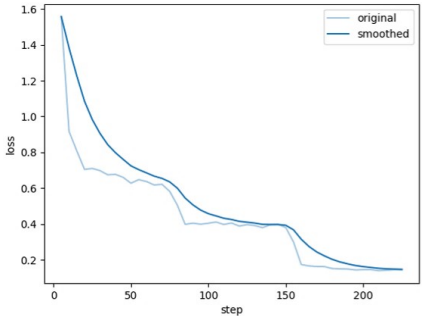

Figure 1: SFT training curve of Qwen2.5-VL-7B on EmbRACE-3K. The smoothed curve shows consistent convergence.

Over the course of 200 training steps, the loss drops from an initial value above 1.5 to below 0.2. The sharp decline in the early phase (0–50 steps) reflects rapid adaptation to the format and structure of instruction-action pairs. Between steps 50 and 150, the curve flattens into a gradual descent, suggesting progressive refinement of action alignment and reasoning consistency. After step 150, the model enters a stable convergence phase, where improvements in loss are marginal, and most trajectories are predicted with high confidence. This progression illustrates the effectiveness of EmbRACE-3K in providing dense and learnable supervision for embodied instruction following.

3.2 RL Training Curve

Figure 2 illustrates the training curves of the Qwen2.5-VL-7B model after undergoing EmbRACE-3K SFT and further optimization through the GRPO algorithm during the reinforcement learning process. In Figure 2(a), we present the defined format training curve, where the reasoning process is encapsulated within the <think></think> tags and the final action decision is enclosed within the <action></action> tags. Due to the model's strong inherent instruction-following capabilities,

the format reward consistently remains close to a high score of 1. Figure 2(b) demonstrates the accuracy of the model's action decisions during training. Over the course of 500 steps, the average accuracy reward improves from below 0.5 to approximately 0.6, showcasing the effectiveness of the reinforcement learning algorithm in enhancing the model's reasoning and decision-making abilities in embodied environments beyond its SFT baseline. Finally, Figure 2(c) depicts the average output length of the model, which remains relatively stable throughout training. This stability reflects the nature of the agent's reasoning, which primarily involves spatial and planning-related tasks, where concise and purposeful reasoning suffices to support effective decision-making.

3.3 Visualization Results of Different Models on EmbRACE-3K

Exploration Tasks. Figures 13 through 15 illustrate model behaviors on the exploration task: "Go find the refrigerator." In this scenario, the target object is not initially visible and requires the agent to actively search by leveraging spatial cues and egocentric observations. Gemini-2.5-Pro and GPT-4o both begin with reasonable initial moves, identifying nearby doors and exploring adjacent rooms. However, both models exhibit repetitive turning patterns after failing to locate the refrigerator within a few steps. Gemini falls into prolonged oscillation, executing many redundant turns without converging toward new areas. GPT-4o performs slightly better by opening a door and entering another room, but ultimately fails to locate the refrigerator and times out without completing the task. In contrast, the Qwen2.5-VL-7B model fine-tuned on EmbRACE-3K demonstrates a more structured and consistent search behavior. It executes a smooth scan of the environment, identifying the refrigerator as soon as it becomes visible. Once detected, it adjusts its heading to center the target and proceeds directly toward it. This behavior reflects improved visual persistence, spatial awareness, and efficient exploration planning—key benefits introduced by the structured task supervision in our dataset.

Dynamic Spatial-Semantic Tasks. Figures 16 through 18 visualize model responses to the instruction "Go to the second pillar on the left in the initial view," a task requiring the agent to interpret ordinal and directional references grounded in the initial egocentric frame. Both Gemini-2.5-Pro and GPT-4o initially identify the general area of the pillars and begin forward motion. However, as they approach the scene, both models exhibit difficulty maintaining the spatial semantics of the instruction. Gemini becomes disoriented near the first pillar, entering a loop of repeated turning without making progress toward the second. GPT-4o initially moves toward the pillars, but after some turns loses alignment with the target and begins a sequence of corrective but ultimately inefficient reorientations. In both cases, the models fail to preserve the mapping between initial-frame spatial language (e.g., "second on the left") and the dynamically evolving viewpoint. By contrast, the Qwen2.5-VL-7B model fine-tuned on EmbRACE-3K maintains a consistent interpretation of the ordinal reference. It accurately identifies the second pillar from the initial view, tracks it through egocentric movement, and successfully approaches it without losing spatial context. This behavior suggests that our dataset's explicit supervision of target reasoning and spatial grounding helps mitigate semantic drift as the agent's perspective changes.

Multi-stage Tasks. Figures 19 through 21 illustrate the performance of Gemini-2.5-Pro, GPT-4o, and Qwen2.5-VL-7B on the same multi-stage task: "First, walk to the left pillar, then go to the right pillar." Both Gemini and GPT-4o are able to complete the first subgoal and reach the left pillar. However, they fail to effectively transition to the second target. Gemini enters prolonged turning loops after reaching the first pillar, issuing multiple orientation commands without successfully identifying or approaching the right pillar. GPT-4o similarly fails to relocate the second target and issues repeated `MidwayTarget` signals even after subgoal completion, indicating confusion or redundancy in goal grounding. In both cases, the task ultimately times out. In contrast, the Qwen2.5-VL-7B model fine-tuned on EmbRACE-3K successfully completes both subgoals in sequence. After reaching the left pillar, it correctly emits a `MidwayTarget` signal and initiates a turn toward the right pillar. Although the second subgoal is temporarily out of view, the model is able to infer its likely direction based on its position in the initial frame. It then proceeds with decisive and directed movement, completing the task without redundant actions. This indicates that the model not only retains the high-level task structure but also leverages earlier visual context to guide planning beyond the current field of view. Such behavior highlights the benefit of fine-tuning on EmbRACE-3K, which supports step-level supervision and reinforces multi-stage goal reasoning under egocentric constraints.

4 Limitations and Broader Impacts

Limitations. Although EmbRACE-3K covers a wide range of embodied reasoning scenarios, its current interactive capabilities are limited to two action types: opening doors and picking or dropping objects. This constraint stems from the capabilities of the underlying UnrealCV-Zoo platform [3], which currently supports only a restricted set of interactions. Expanding the set of supported object manipulations, tool use, and environment-triggered events is a promising direction for future extensions. Additionally, while our dataset emphasizes step-wise reasoning and egocentric visual grounding, it does not yet include human-agent interaction or multi-agent coordination, which are increasingly important in real-world embodied AI applications.

Broader impacts. EmbRACE-3K contributes to the development of vision-language agents that reason and act in physically grounded, sequential decision-making tasks. By providing a benchmark focused on closed-loop interaction, it encourages research that moves beyond passive scene understanding toward actionable multimodal intelligence. The dataset does not contain real-world images or identifiable human data, minimizing risks related to privacy or fairness. However, as embodied agents become more capable, their deployment in physical or virtual environments raises questions of safety, transparency, and alignment with user intent. We encourage the community to use EmbRACE-3K in ways that promote interpretable, controllable, and socially beneficial AI systems.

Objective: Design navigation tasks for embodied AI training in Unreal Engine.
The objects within 1000-meter are: [{', '.join(nearby_object_names)}]

Core Challenges with Examples
Challenge 1: Short-sighted Exploration
- For situations where the agent cannot see the target from the first-person view initially and has no knowledge of its location, the model may fail to produce continuous, sensible decisions.
- Example: “Explore the surroundings and walk to the red car.” A conventional model might turn left once, see no red car, then turn back immediately; then turn right once, see no red car, and turn back. Seeing no red car on either side, it simply moves forward. However, the red car might require turning multiple times to discover.

Challenge 2: Dynamic Spatial-Semantic Drift
- Instructions like “Go to the door of the white house in front of you” or “Go to the second intersection” can become invalid as the agent moves, because references such as “in front of you” and “the second intersection” change relative to the agent’s new position.

Challenge 3: Target Forgetting
- In multi-stage tasks such as “First walk to the trash can, then walk to the red car,” the agent may forget the second target after completing the first sub-task.
- Even in single-step tasks like “Walk near the red car,” the agent might momentarily lose sight of the car if it takes a large stride and fail to readjust its path, indicating insufficient long-term target binding in a dynamic environment.

Task Design Rules
1. Half of the instructions should describe simple, direct paths to clearly visible objects.
2. Avoid any references to previously seen objects or prior knowledge (no memory of earlier views).
3. For exploration tasks, assume the agent has only its current first-person view with no map priors.
4. Each command should be concise and clear (e.g., “Go to the red car”).
5. All commands must be in English.
6. Every command must specify a final destination (e.g., “Go to X”), not single-step actions like “turn left.”
7. Ensure no ambiguity: there should not be multiple objects in the scene that match the same description.
8. For large objects, specify a distinct component (e.g., "go to the door of the house" instead of "go to the house").
9. For dynamic spatial semantics tasks, do not include targets that are not visible in the current first-person view (e.g., avoid “Go to the main door behind you”).

Output Format Rules
1. Return exactly 10 tasks as a list of dictionaries in JSON format. No other extra information.
2. Each dictionary must contain:
  - 'Instruction': Full navigation command
  - 'Target Object': List of EXACT name(s) from the objects (within 1000-meter) list.
    - Single stage: Single-element list (e.g., ["A"])
    - Multiple stage: multi-element ordered list (e.g., ["A", "B"])
  - 'Reason': Explain why this task is relevant or interesting
  - 'Type': Integer code:
    - 0: Basic (direct path to visible target)
    - 1: Exploration (requires visual search)
    - 2: Dynamic spatial semantic
    - 3: Multi-stage (only two sequential sub-goals here)

Strict Rule (Must Be Followed)
The **input image** is the agent’s only **first-person egocentric view**. It is the sole basis for determining what the agent can currently see.
Although all candidate objects are within a 1000-meter radius, only those that actually appear in the input image are considered **visible to the agent**. Any object that does **not appear** in the image must be treated as **invisible**, regardless of distance.

Visibility constraints by task type
- **Type 0 (Basic):** Target must be clearly visible and directly reachable in the input image.
- **Type 1 (Exploration):** Target is currently invisible (not shown in the input image) and must be discovered through navigation.
- **Type 2 (Dynamic Spatial Semantic):** Target is visible, but requires spatial or ordinal reasoning to identify (e.g., “the second chair,” “the one on the right,” “the front house”).
- **Type 3 (Multi-stage):** Both targets must be visible in the input image. The challenge lies in remembering or reasoning about the second target after it temporarily goes out of view.

Do **not** use any object that is outside the input image for Type 0, Type 2, or Type 3 tasks. Only visible objects should be used for these task types.

Figure 3: Prompt template used to generate navigation task instructions with Gemini. The prompt defines core challenges (short-sighted exploration, spatial-semantic drift, target forgetting), formalizes instruction types (Basic, Exploration, Spatial Semantic, Multi-stage), and enforces strict visibility constraints based on the agent’s egocentric view. Tasks are designed to simulate realistic, instruction-guided navigation in interactive embodied settings.

You are an **embodied AI agent** making real-time decisions during an embodied task.
The following is a sequence of first-person images and actions, recorded in Unreal Engine. Each image represents your visual observation just before you chose an action.
You may access the full trajectory to help disambiguate scenes, but your explanations must be written as if you do not know what happens next. Never refer to future steps.

Your responsibilities for each step:
1. Articulate your internal reasoning for the chosen action:
   - This is the only section where reasoning and inference are allowed.
   - Think and write as yourself — the embodied agent — deciding how to act based on what you perceive in the scene.
   - Your reasoning must be forward-looking: simulate the thought process that leads to an action based on the current view.
   - Use first-person, natural-sounding language like:
     "I see the barrels on the left-hand side. To center them in my view, I'll turn slightly to the left."
   - Focus on task-relevant objects you perceive in the scene. Mention **all visible objects of the same category as the task target** when relevant to your decision-making, including their spatial location relative to your viewpoint (e.g., "on the left-hand side", "in the center", "on the right-hand side"). Base spatial locations on the image frame being divided vertically into three roughly equal sections: **left-hand side, center, and right-hand side**.
   - Do not use phrasing like "to the left" or "to the right" that implies endpoint targeting. Always describe spatial alignment with "on the left-hand side", "in front", "on the right-hand side", etc.
   - Assume your camera and body are aligned (ego-view equals agent orientation), unless the action is LookUp, LookDown, LookHand, or LookBack.
   - **Crucially, ensure your stated reasoning is geometrically consistent with the spatial locations of objects as observed in the image.** For example, if you perceive an object on the left-hand side, turning left should logically follow if your goal is to center it or move towards it. Turning right would move it further left in your view. Use this geometric logic to ensure your reasoning makes physical sense.
   - If the visual scene seems inconsistent with geometric turning logic based on previous steps, trust the geometric reasoning and assume the visual elements support the action chosen.
   - Always prioritize what makes sense physically.

   Language constraints:
   1. The word "right" may only be used in explicit spatial phrases (e.g., "on the right-hand side", "in the right half of the image").
      - You must **never** use "right" in a non-spatial way (e.g., "right beside me", "right away", "the right option").
   2. If the task instruction contains a phrase like "on the right", you must:
      - Put that phrase in quotation marks: "on the right".
      - Explicitly state that this is a quote from the instruction.
      - If the object's position has changed relative to your view, explain this shift:
        Example: The instruction says "on the right", but the object is now on the left-hand side of my view.

2. If the task instruction includes multiple stages (e.g., "first go to the barrels, then go to the tree"), follow these strict rules:
   - Before the appearance of an action tagged with "MidwayTarget", you must **only consider the first-stage goal** in your reasoning.
   - Do not mention, plan, or hint at the second-stage goal before MidwayTarget.
   - When the action is MidwayTarget, **only explain why the first-stage goal is satisfied**.
   - You may begin considering the second-stage goal only in the frame **after** the MidwayTarget action.

Task instruction:
{instruction}

Trajectory steps:
{stepwise_text}

Return a JSON list, one entry per frame. Each entry must include:

- "image_name": the frame's image filename (e.g., "002.jpg")
- "action": the action you chose after viewing this frame
- "thinking": your internal reasoning for the action
  - This is the only field where inference is allowed.
  - Think in first-person, based on current and past frames only.
  - Directly reflect your visual perception of the scene, focusing on task-relevant objects and their spatial locations (left/center/right).
  - Ensure reasoning is **consistent with perceived spatial location** and geometric turning logic.
  - Avoid any mention of second-stage goals unless MidwayTarget has occurred.
  - If MidwayTarget has occurred, you may begin reasoning toward the second-stage goal.

Begin your step-by-step reasoning now.

Figure 4: Prompt template used to generate step-wise reasoning annotations with Gemini-2.5-Pro. The input includes the task instruction, the complete sequence of egocentric visual observations, and the full history of executed actions. At each step, the model generates a first-person explanation for the current action, grounded in spatial layout and task semantics, while explicitly avoiding any reference to future states.

```
[
  ......
  {
    "messages": [
      {
        "role": "user",
        "content": "<image>\nYou are an autonomous embodied agent operating in Unreal Engine.\nYour goal is to select the next best action that helps efficiently and safely complete the assigned task.\n\nInputs:\n- TASK: \"Survey the environment for any door, then approach and open it.\"\n- FIRST-PERSON VIEW IMAGES: A time-ordered sequence of the agent's first-person views. Images later in the sequence correspond to later time steps.\n- TRAJECTORY: []\n\nAnalysis Criteria:\n1. Trajectory Evaluation:\n- Detect repeated movement patterns such as loops or oscillations\n- Identify inefficiencies like back-and-forth motion\n- Infer how the agent's relative position to the target has evolved\n\n2. Current Visual State:\n- Identify visible obstacles around the agent\n- Estimate the proximity of the target object based on size/position\n- Evaluate navigable movement directions (e.g., clear paths vs. blocked ones)\n\n3. Task-oriented Action Assessment:\n- Consider which next action is most likely to bring the agent closer to completing the task effectively\n- Avoid actions that merely preserve motion without strategic progress\n\nAdditional Objectives:\n- Maximize path efficiency\n- Avoid unproductive patterns: circular loops, repeated back-and-forth movements\n\nOutput MUST be in the following format:\n<think>Your reasoning about the next action</think><action>One of: MoveForward, MoveBackward, TurnLeft, TurnRight, Finish, LookUp, LookDown, LookHand, LookBack, JumpForward, OpenDoor, PickObject, DropObject, MidwayTarget</action>\n\nConstraints:\n- The `action` value must be exactly one of the allowed commands listed above (case-sensitive).\n- Use \"MidwayTarget\" if the agent has just reached a **sub-goal** or **intermediate objective** in a multi-stage task."
      },
      {
        "role": "assistant",
        "content": "<think>My task is to find a door, approach it, and open it. I don't see any doors in my current view, which shows shelves on the right-hand side and tables with cabinets on the left-hand side and in the center. I'll turn left to survey more of the environment.</think><action>TurnLeft</action>"
      }
    ],
    "images": [
      "embrace3k_train_image/SuburbNeighborhood_Day/x03703_y01064_z-0132_door0/000.jpg"
    ]
  },
  {
    "messages": [
      {
        "role": "user",
        "content": "<image>\n<image>\nYou are an autonomous embodied agent operating in Unreal Engine.\nYour goal is to select the next best action that helps efficiently and safely complete the assigned task.\n\nInputs:\n- TASK: \"Survey the environment for any door, then approach and open it.\"\n- FIRST-PERSON VIEW IMAGES: A time-ordered sequence of the agent's first-person views. Images later in the sequence correspond to later time steps.\n- TRAJECTORY: ['TurnLeft']\n\nAnalysis Criteria:\n1. Trajectory Evaluation:\n- Detect repeated movement patterns such as loops or oscillations\n- Identify inefficiencies like back-and-forth motion\n- Infer how the agent's relative position to the target has evolved\n\n2. Current Visual State:\n- Identify visible obstacles around the agent\n- Estimate the proximity of the target object based on size/position\n- Evaluate navigable movement directions (e.g., clear paths vs. blocked ones)\n\n3. Task-oriented Action Assessment:\n- Consider which next action is most likely to bring the agent closer to completing the task effectively\n- Avoid actions that merely preserve motion without strategic progress\n\nAdditional Objectives:\n- Maximize path efficiency\n- Avoid unproductive patterns: circular loops, repeated back-and-forth movements\n\nOutput MUST be in the following format:\n<think>Your reasoning about the next action</think><action>One of: MoveForward, MoveBackward, TurnLeft, TurnRight, Finish, LookUp, LookDown, LookHand, LookBack, JumpForward, OpenDoor, PickObject, DropObject, MidwayTarget</action>\n\nConstraints:\n- The `action` value must be exactly one of the allowed commands listed above (case-sensitive).\n- Use \"MidwayTarget\" if the agent has just reached a **sub-goal** or **intermediate objective** in a multi-stage task."
      },
      {
        "role": "assistant",
        "content": "<think>I've turned left. I now see a wooden panel or wall on the left-hand side, and tables and cabinets in the center and on the right-hand side. I still don't see a clear door. I'll continue turning left to scan the area.</think><action>TurnLeft</action>"
      }
    ],
    "images": [
      "embrace3k_train_image/SuburbNeighborhood_Day/x03703_y01064_z-0132_door0/000.jpg",
      "embrace3k_train_image/SuburbNeighborhood_Day/x03703_y01064_z-0132_door0/001.jpg"
    ]
  },
  ......
]
```


Figure 5: Examples of a training prompt used for supervised fine-tuning with reasoning. The input includes the task instruction, a full history of egocentric observations and actions, and the model is expected to output both a natural language explanation (`<think>`) and the next action (`<action>`).

```
[
  ......
  {
    "messages": [
      {
        "role": "user",
        "content": "<image>\nYou are an autonomous embodied agent operating in Unreal Engine.\nYour goal is to select the next best action that helps efficiently and safely complete the assigned task.\n\nInputs:\n- TASK: \"Survey the environment for any door, then approach and open it.\"\n- FIRST-PERSON VIEW IMAGES: A time-ordered sequence of the agent's first-person views. Images later in the sequence correspond to later time steps.\n- TRAJECTORY: []\n\nAnalysis Criteria:\n1. Trajectory Evaluation:\n- Detect repeated movement patterns such as loops or oscillations\n- Identify inefficiencies like back-and-forth motion\n- Infer how the agent's relative position to the target has evolved\n\n2. Current Visual State:\n- Identify visible obstacles around the agent\n- Estimate the proximity of the target object based on size/position\n- Evaluate navigable movement directions (e.g., clear paths vs. blocked ones)\n\n3. Task-oriented Action Assessment:\n- Consider which next action is most likely to bring the agent closer to completing the task effectively\n- Avoid actions that merely preserve motion without strategic progress\n\nAdditional Objectives:\n- Maximize path efficiency\n- Avoid unproductive patterns: circular loops, repeated back-and-forth movements\n\nOutput MUST be in the following format:\n<action>One of: MoveForward, MoveBackward, TurnLeft, TurnRight, Finish, LookUp, LookDown, LookHand, LookBack, JumpForward, OpenDoor, PickObject, DropObject, MidwayTarget</action>\n\nConstraints:\n- The `action` value must be exactly one of the allowed commands listed above (case-sensitive).\n- Use \"MidwayTarget\" if the agent has just reached a **sub-goal** or **intermediate objective** in a multi-stage task."
      },
      {
        "role": "assistant",
        "content": "<action>TurnLeft</action>"
      }
    ],
    "images": [
      "embrace3k_train_image/SuburbNeighborhood_Day/x03703_y01064_z-0132_door0/000.jpg"
    ]
  },
  {
    "messages": [
      {
        "role": "user",
        "content": "<image>\n<image>\nYou are an autonomous embodied agent operating in Unreal Engine.\nYour goal is to select the next best action that helps efficiently and safely complete the assigned task.\n\nInputs:\n- TASK: \"Survey the environment for any door, then approach and open it.\"\n- FIRST-PERSON VIEW IMAGES: A time-ordered sequence of the agent's first-person views. Images later in the sequence correspond to later time steps.\n- TRAJECTORY: ['TurnLeft']\n\nAnalysis Criteria:\n1. Trajectory Evaluation:\n- Detect repeated movement patterns such as loops or oscillations\n- Identify inefficiencies like back-and-forth motion\n- Infer how the agent's relative position to the target has evolved\n\n2. Current Visual State:\n- Identify visible obstacles around the agent\n- Estimate the proximity of the target object based on size/position\n- Evaluate navigable movement directions (e.g., clear paths vs. blocked ones)\n\n3. Task-oriented Action Assessment:\n- Consider which next action is most likely to bring the agent closer to completing the task effectively\n- Avoid actions that merely preserve motion without strategic progress\n\nAdditional Objectives:\n- Maximize path efficiency\n- Avoid unproductive patterns: circular loops, repeated back-and-forth movements\n\nOutput MUST be in the following format:\n<action>One of: MoveForward, MoveBackward, TurnLeft, TurnRight, Finish, LookUp, LookDown, LookHand, LookBack, JumpForward, OpenDoor, PickObject, DropObject, MidwayTarget</action>\n\nConstraints:\n- The `action` value must be exactly one of the allowed commands listed above (case-sensitive).\n- Use \"MidwayTarget\" if the agent has just reached a **sub-goal** or **intermediate objective** in a multi-stage task."
      },
      {
        "role": "assistant",
        "content": "<action>TurnLeft</action>"
      }
    ],
    "images": [
      "embrace3k_train_image/SuburbNeighborhood_Day/x03703_y01064_z-0132_door0/000.jpg",
      "embrace3k_train_image/SuburbNeighborhood_Day/x03703_y01064_z-0132_door0/001.jpg"
    ]
  },
  ......
]
```


Figure 6: Examples of a training prompt used for supervised fine-tuning or reinforcement learning without reasoning. The model receives the task instruction, full visual history, and past actions, and directly predicts the next action in `<action>` format.

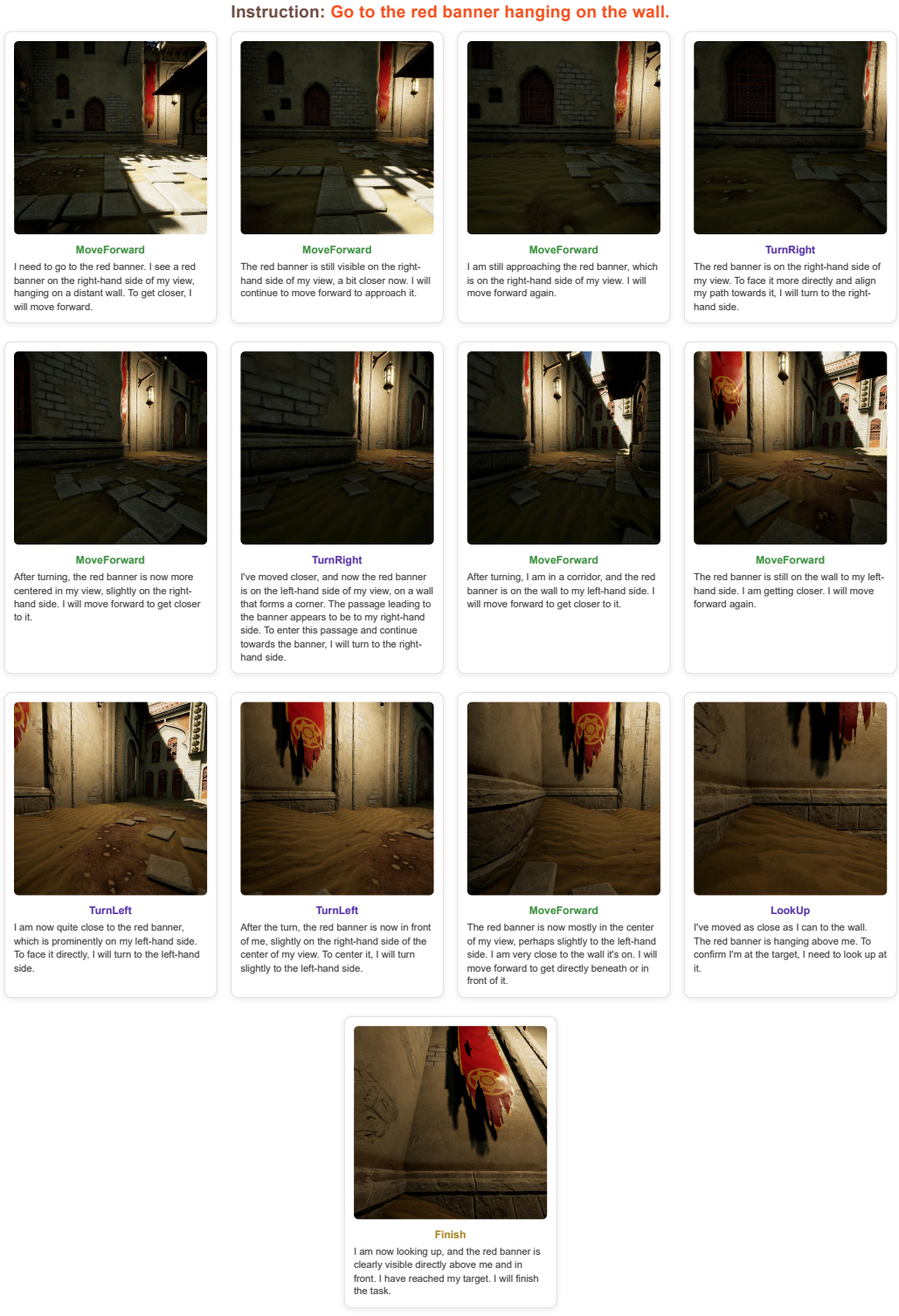

Figure 7: Visualization of a **Basic** task. The target object is clearly visible from the start, and the agent moves directly toward it with minimal ambiguity. Reasoning focuses on spatial orientation and efficient path alignment.

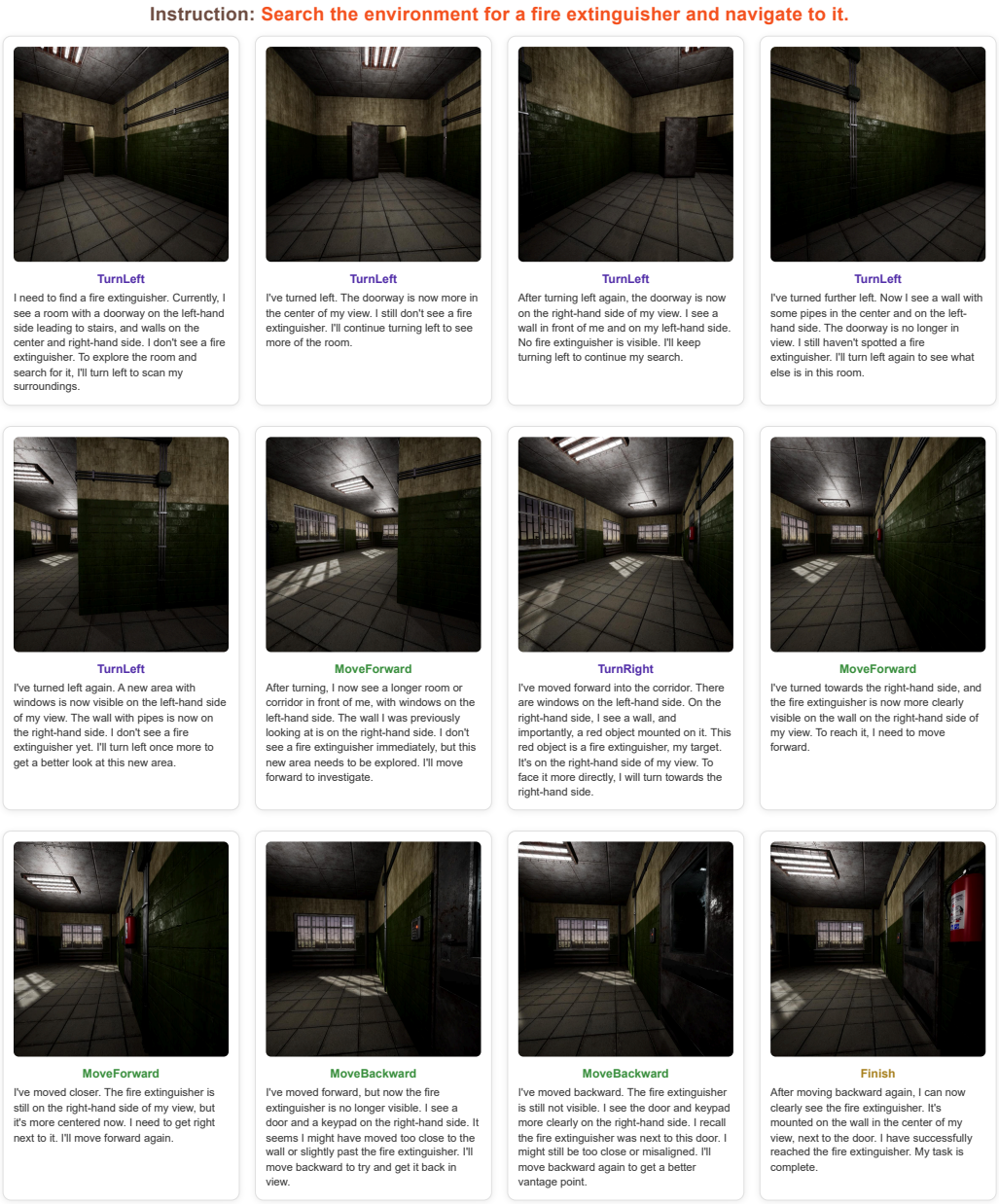

Figure 8: Visualization of an **Exploration** task. The target is initially out of view, requiring the agent to conduct visual search by turning and scanning the scene before identifying and approaching the goal.

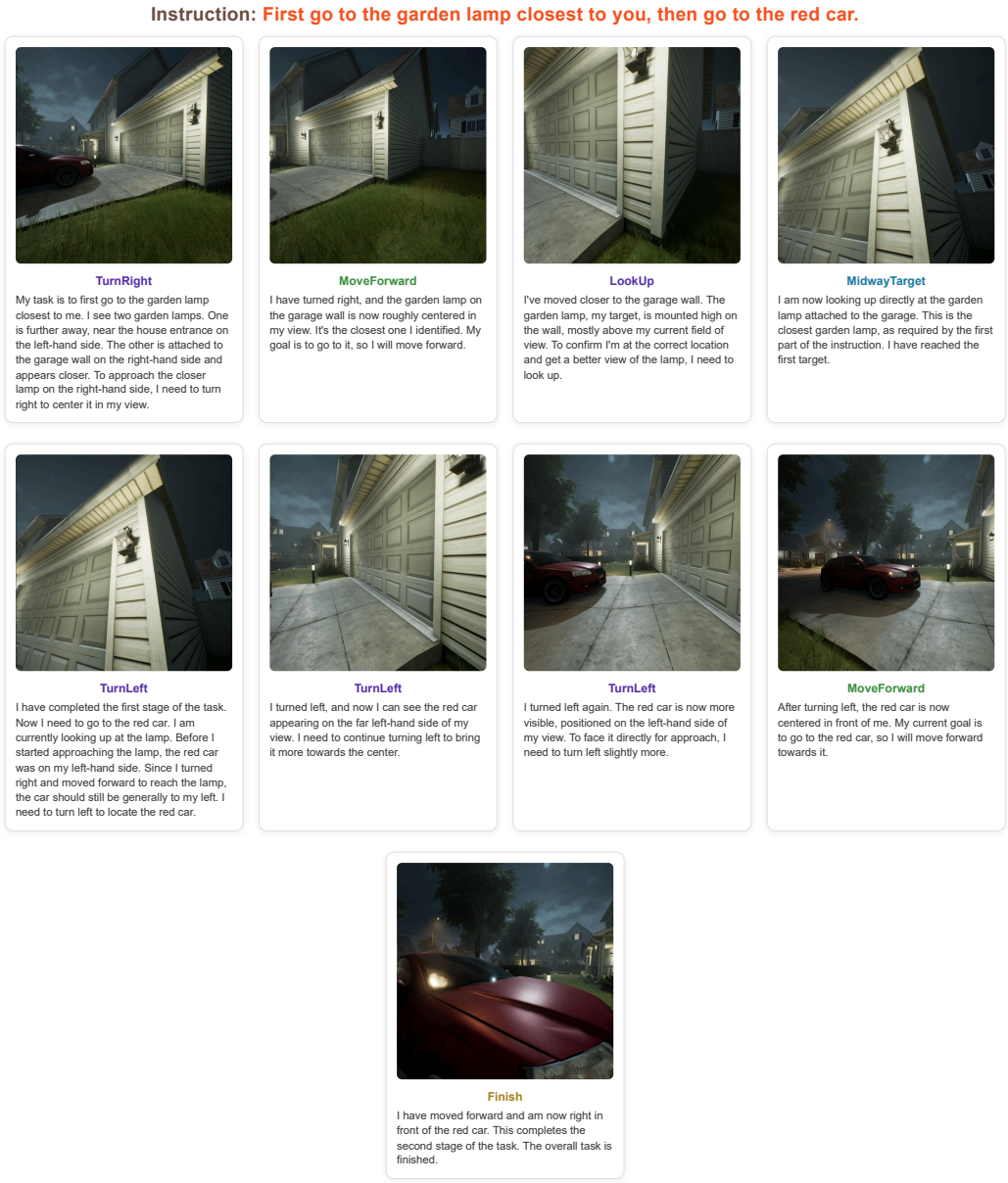

Figure 9: Visualization of a **Multi-stage** task. The agent must complete a sequence of subgoals in order. The reasoning must shift at the appropriate time, and target persistence is required to recall goals after partial occlusion or disorientation.

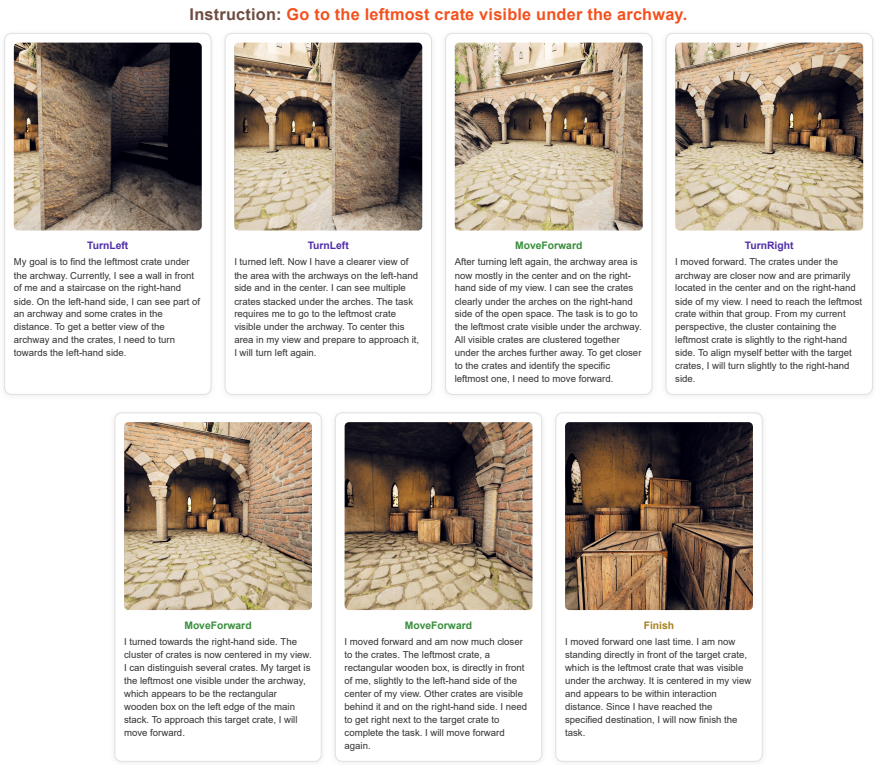

Figure 10: Visualization of a **Dynamic Spatial-Semantic** task. The instruction involves relational spatial references such as "leftmost" and "under the archway", which require the agent to resolve dynamic spatial semantics as its viewpoint evolves.

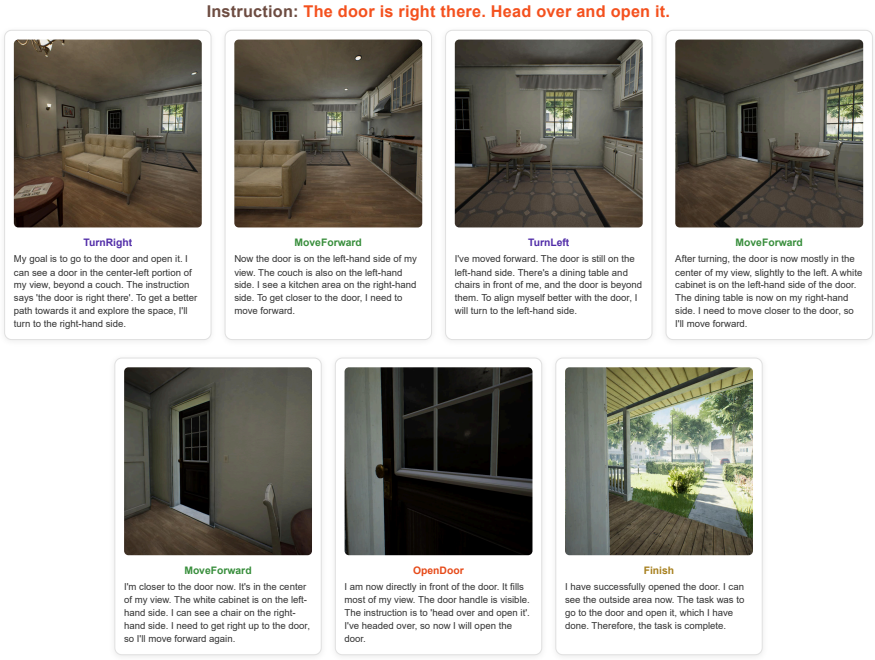

Figure 11: Visualization of an **Interaction: Open Door** task. The agent must navigate to a nearby visible door and explicitly trigger an `OpenDoor` action. Reasoning must ensure spatial alignment with the door and recognize successful interaction.

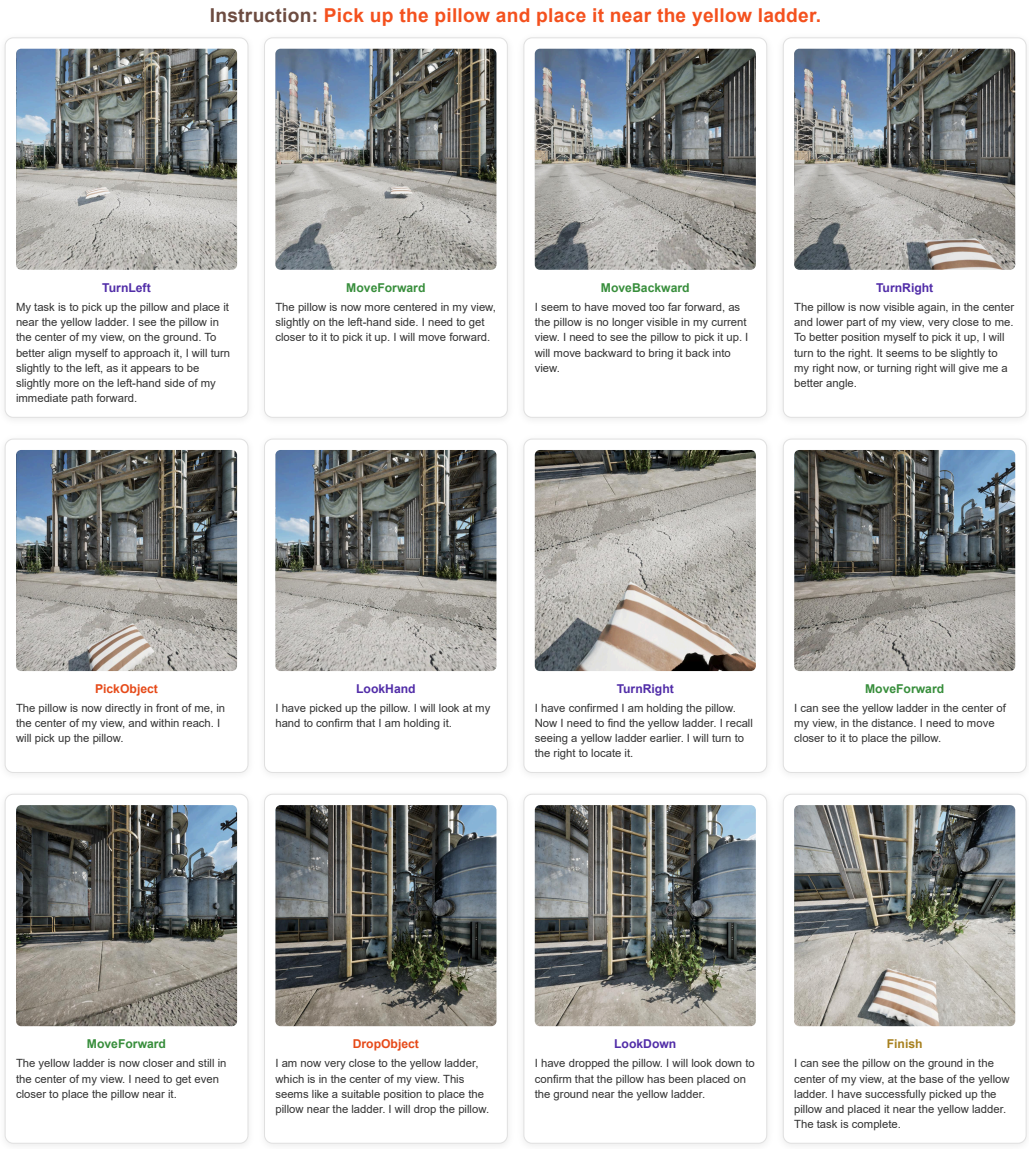

Figure 12: Visualization of an **Interaction: Pick and Drop** task. The agent must locate a manipulable object, perform a pick-up, navigate to a reference location, and execute a drop. The reasoning combines manipulation awareness with spatial goal alignment.

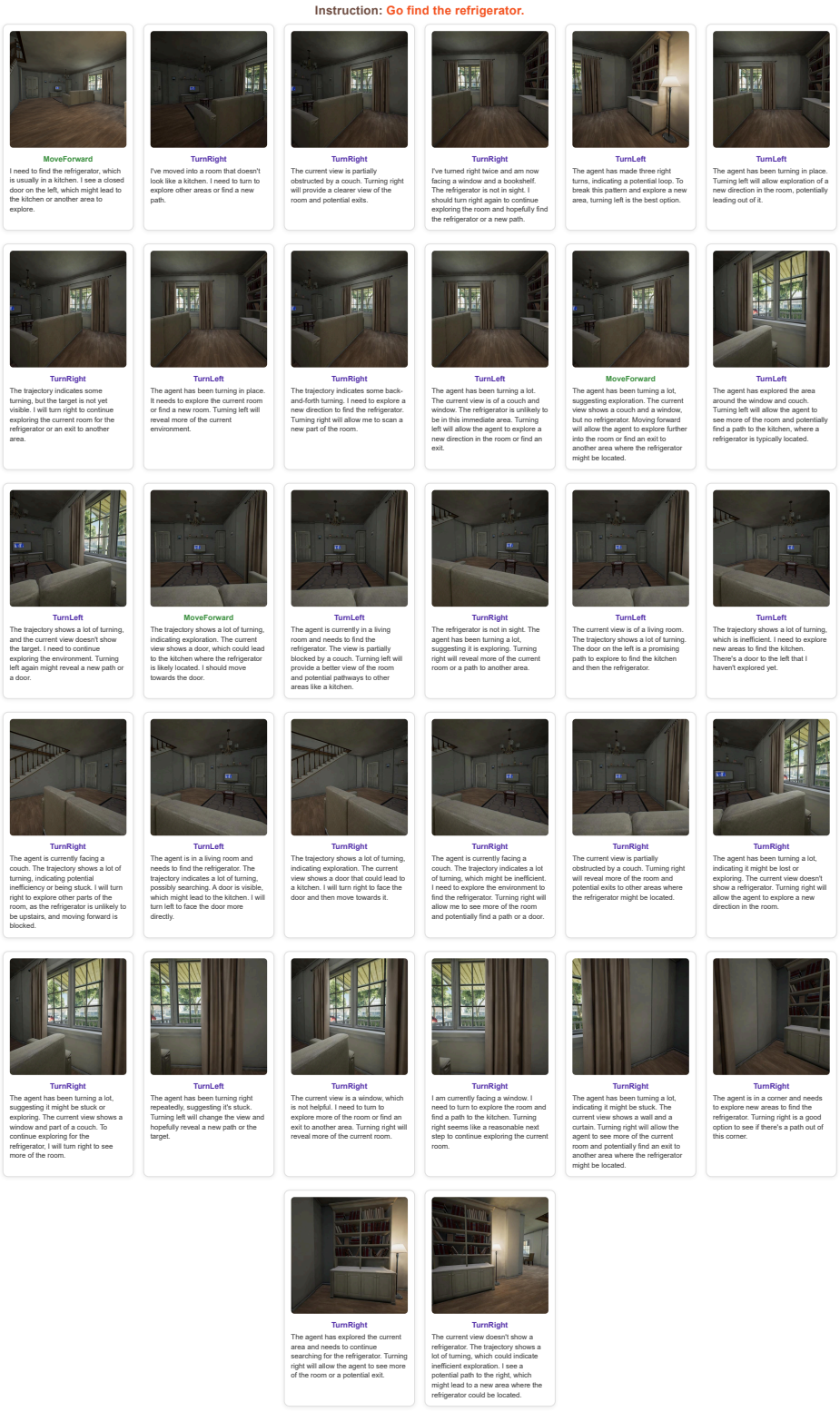

Figure 13: Exploration task example for Gemini-2.5-Pro. The model fails to locate the refrigerator and enters a loop of repeated turns.

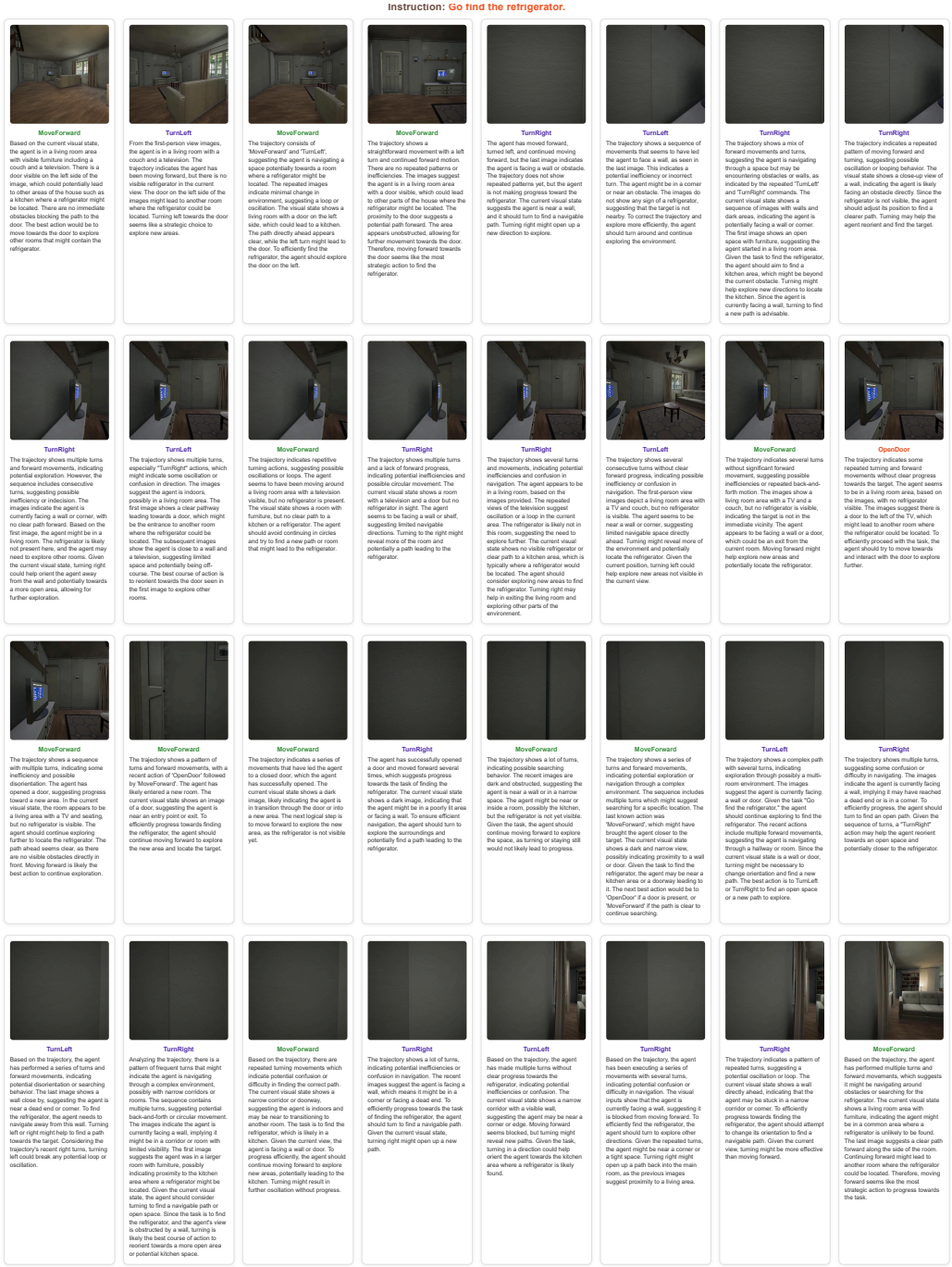

Figure 14: Exploration task example for GPT-4o. The model explores multiple rooms but fails to find the refrigerator and times out.

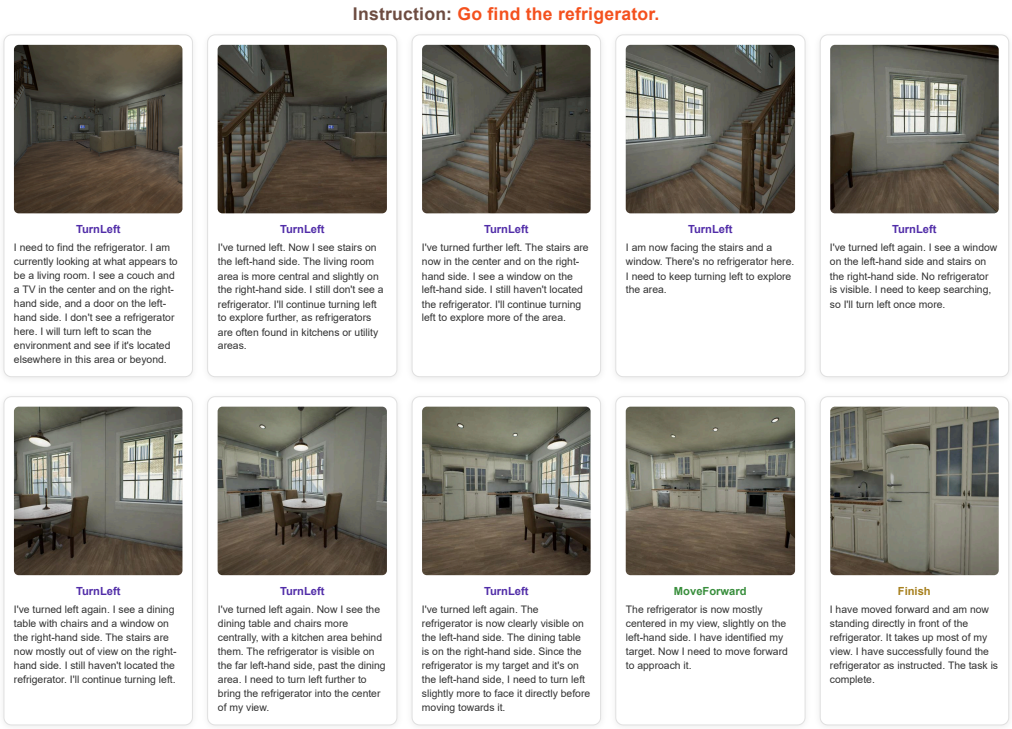

Figure 15: Exploration task example for Qwen2.5-VL-7B fine-tuned on EmbRACE-3K. The model performs a full scan, identifies the refrigerator, and completes the task.

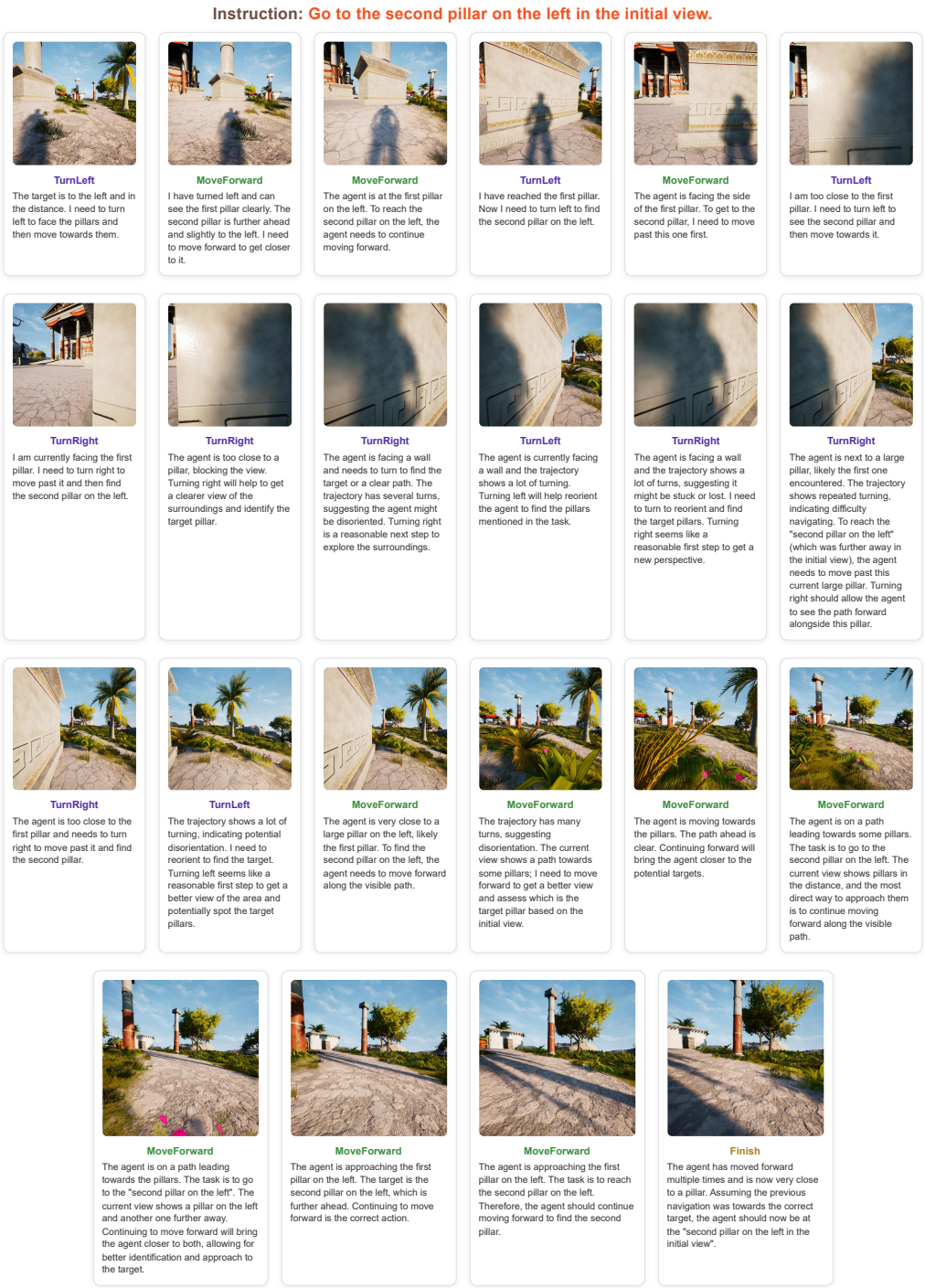

Figure 16: Dynamic spatial-semantic task example for Gemini-2.5-Pro. The model fails to maintain reference alignment and becomes disoriented near the first pillar.

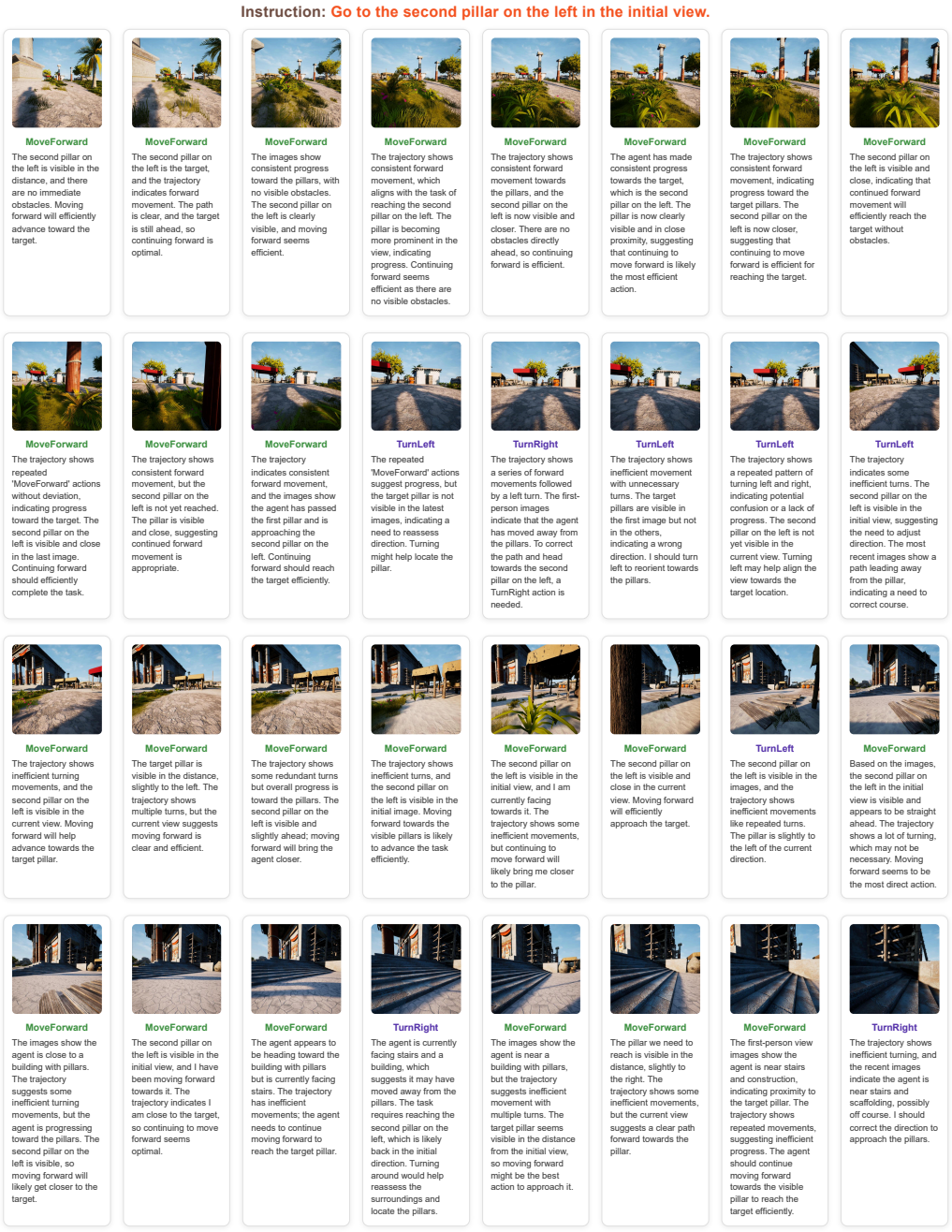

Figure 17: Dynamic spatial-semantic task example for GPT-4o. The model initially heads toward the correct region but fails to preserve ordinal reference during task.

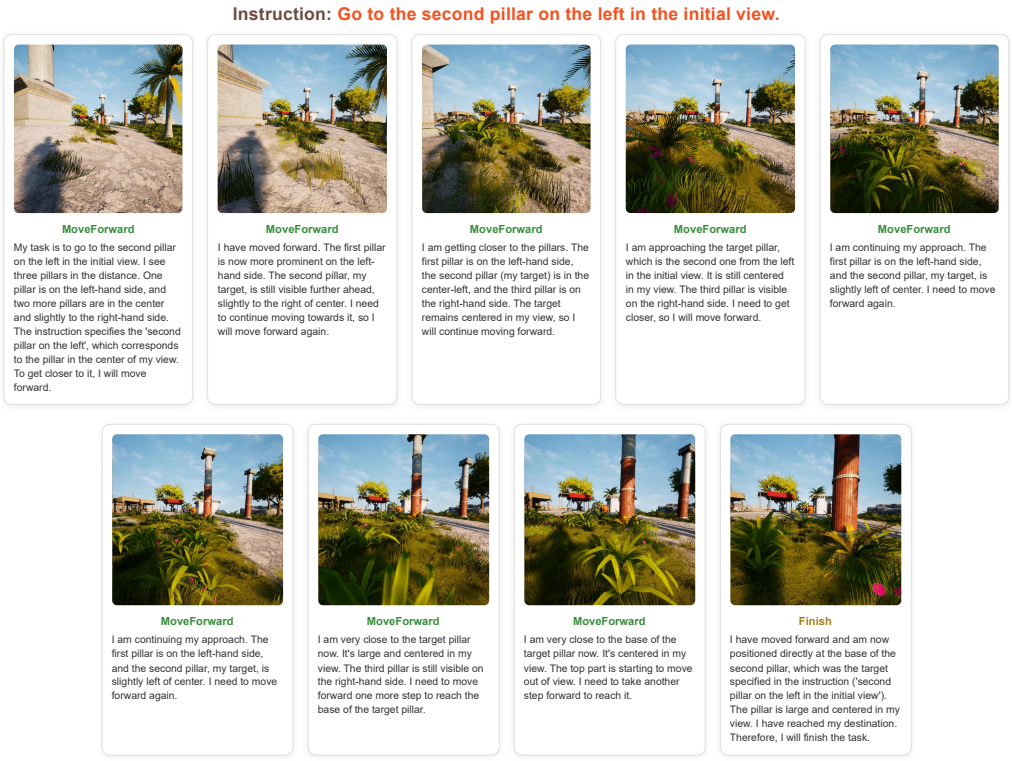

Figure 18: Dynamic spatial-semantic task example for Qwen2.5-VL-7B fine-tuned on EmbRACE-3K. The model correctly interprets and tracks the second pillar through egocentric perspective changes.

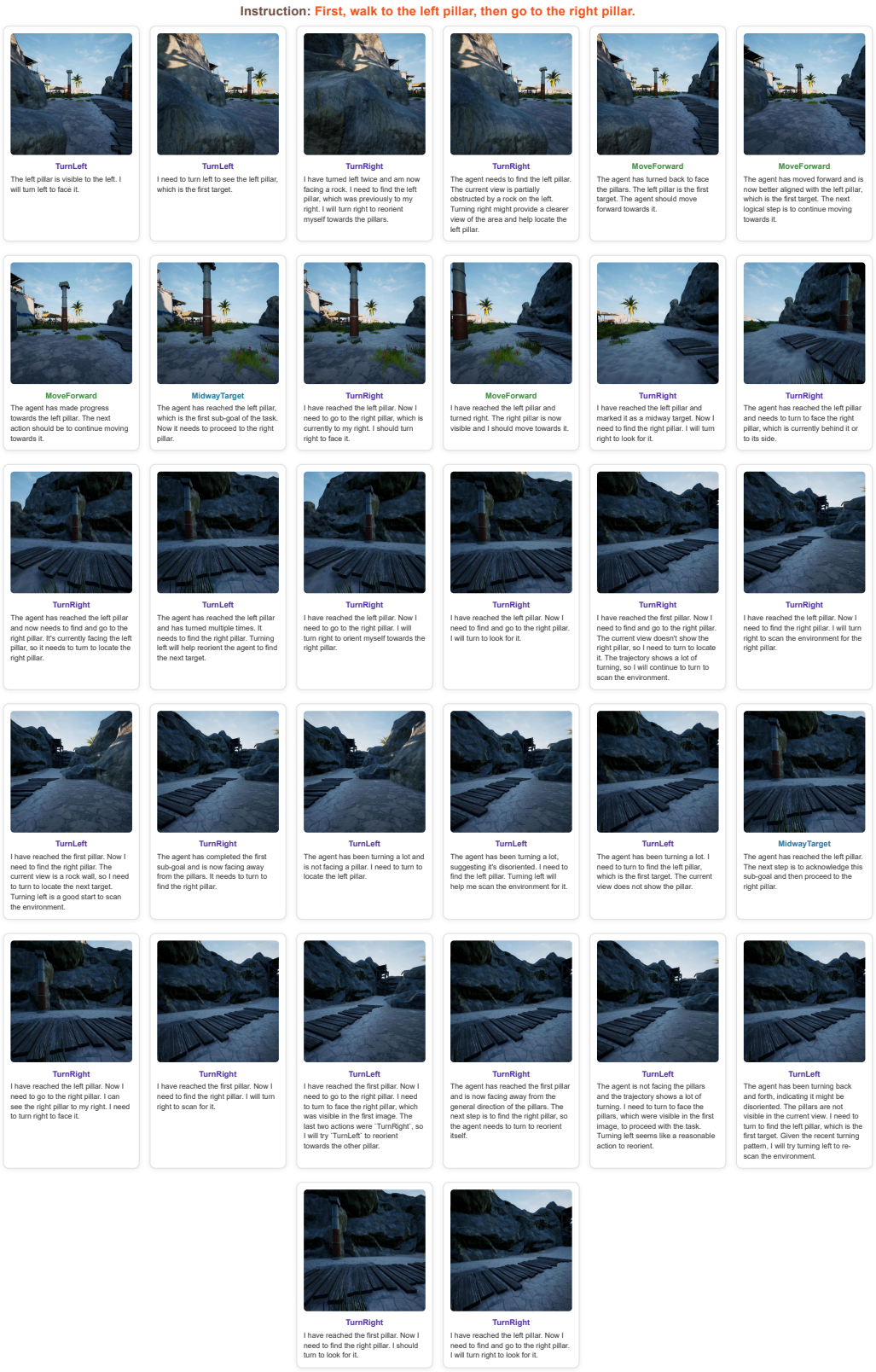

Figure 19: Multi-stage task visualization for Gemini-2.5-Pro. The model reaches the first subgoal but fails to localize the second, exhibiting repeated turning and no successful task completion.

Figure 20: Multi-stage task visualization for GPT-4o. The model completes the first-stage goal but fails to relocate the second, with oscillatory behavior and multiple redundant midway signals.

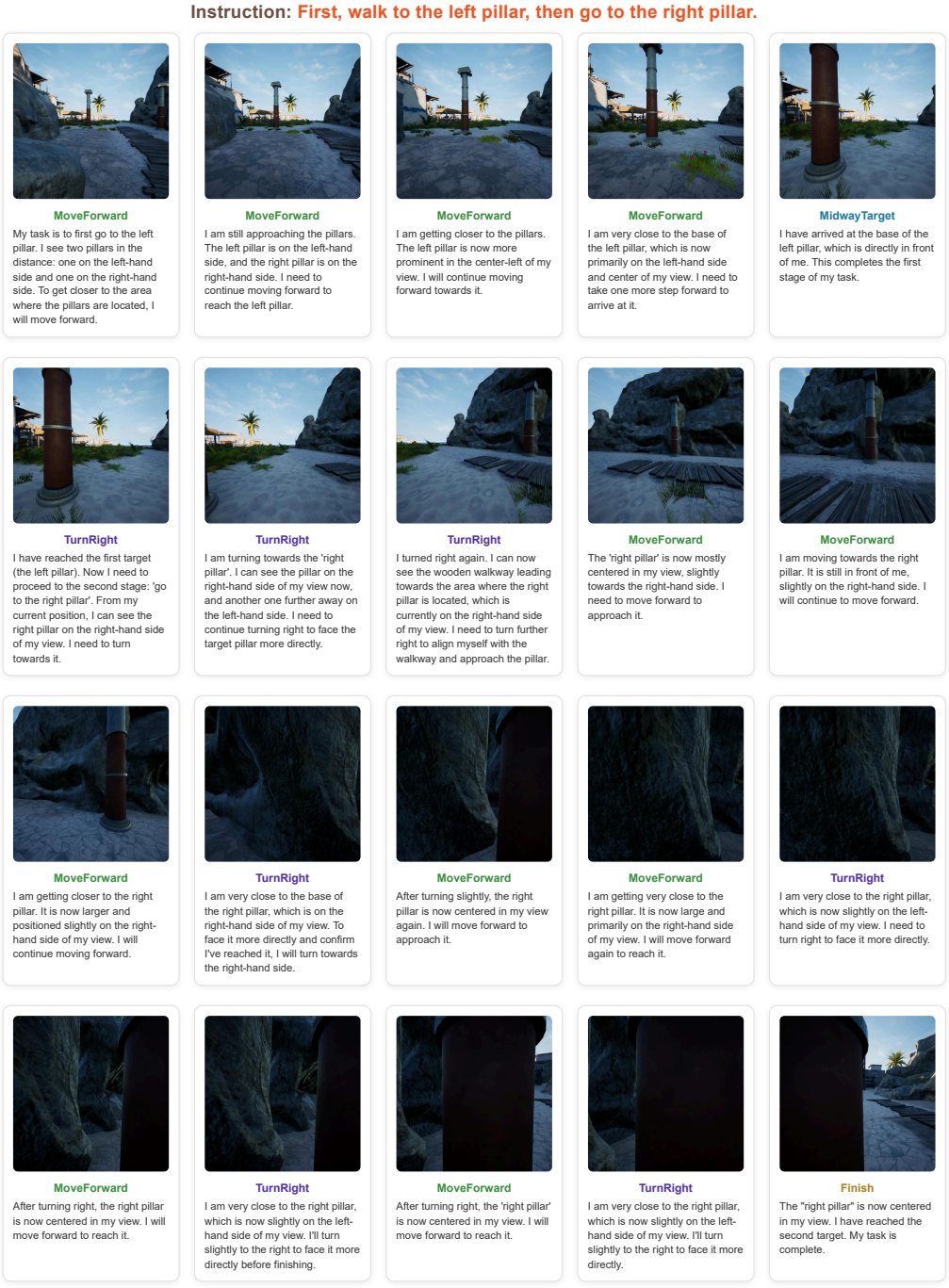

Figure 21: Multi-stage task visualization for Qwen2.5-VL-7B fine-tuned on EmbRACE-3K. The model successfully completes both subgoals with coherent orientation and target tracking.

References

[1] Shuai Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, Sibo Song, Kai Dang, Peng Wang, Shijie Wang, Jun Tang, et al. Qwen2. 5-vl technical report. *arXiv preprint arXiv:2502.13923*, 2025.

[2] DeepMind. Gemini 1.5: Unlocking multimodal understanding across millions of tokens. *arXiv preprint arXiv:2403.05530*, 2024.

[3] Fangwei Zhong, Kui Wu, Churan Wang, Hao Chen, Hai Ci, Zhoujun Li, and Yizhou Wang. Unrealzoo: Enriching photo-realistic virtual worlds for embodied ai. *arXiv preprint arXiv:2412.20977*, 2024.