

STRIDER: Navigation via Instruction-Aligned Structural Decision Space Optimization

Diqi He^{1*}, Xuehao Gao^{1*}, Hao Li^{1,2}, Junwei Han^{1,3}, Dingwen Zhang^{1†}

¹Northwestern Polytechnical University

²Nanyang Technological University

³Chongqing University of Posts and Telecommunications

<https://github.com/diqihe666/STRIDER-Nav>

Supplemental Material

The supplemental material is organized as follows:

- **A Implementation Details:** implementation details of STRIDER, including the system hyperparameters, prompting setup, and deployment configurations.
- **B Robustness Analysis:** robustness analysis under two settings: perceptual degradation and mid-trajectory perturbation.
- **C Visualization Results:** qualitative visualization results, including a step-by-step case study, additional episodes, and failure cases.
- **D Limitations and Future Work:** discussion of current limitations related to perception settings and foundation model dependencies, along with future directions.
- **E Broader Impact:** broader impact analysis concerning the societal implications of deploying large-scale VLM- and LLM-based navigation systems.

These sections provide further insights into our system design, experimental robustness, and qualitative behaviors, along with reflections on limitations and potential societal implications.

A Implementation Details

In this section, we provide comprehensive implementation details of our proposed STRIDER. We begin by outlining the overall system pipeline and deployment setup, including the perception configuration, structured waypoint generation, task-alignment mechanism, and LLM-based planning loop. We then describe the design of our prompting templates for the Vision-Language Model (VLM), which play a central role in both environmental perception and progress feedback.

A.1 System Pipeline and Deployment Settings

All experiments were conducted in a simulated VLN-CE environment using the Matterport3D simulator. We followed the standard evaluation protocols defined by the R2R-CE and RxR-CE benchmarks.

Perception Setup. At each timestep t , the agent receives a 360° RGB-D panoramic observation consisting of 12 RGB-D frames spaced at 30° intervals. All frames are captured at a fixed horizontal elevation, without looking above or below eye level. The RGB images are resized to $244 \times 244 \times 3$,

*Equal contribution.

†Corresponding author.

and the depth maps are resized to 256×256 . To obtain a holistic understanding of the surrounding environment, we concatenate the 12 RGB frames into a single 3×4 grid image and input it into the Vision-Language Model (VLM) to generate a global scene description. This allows the agent to build a comprehensive understanding of the overall context. Additionally, for each candidate waypoint, we feed the corresponding directional RGB view into the model to obtain a localized visual description. We adopt Qwen-VL-Max (accessed via API) as the default VLM for all perception and feedback-related tasks in our system.

Structured Waypoint Generation. We extract spatially structured waypoints by processing the local point cloud centered on the agent. Points with vertical height below $\delta_h = -1.0\text{m}$ are retained to approximate surrounding static structures. These points are projected onto the x - z plane to form a 2D occupancy map. To obtain a clean and geometrically consistent traversable area, we apply contour filling followed by Gaussian smoothing (kernel size 75×75), which helps simplify irregular boundaries and suppress small artifacts, thereby reducing unnecessary branches in the subsequent skeleton. The result is then binarized and filtered to retain only the largest connected region. We extract a topological skeleton via morphological thinning and identify endpoints as pixels with degree = 1 in the 8-connected neighborhood, corresponding to path-extremities in the navigable space. These endpoints are merged within a 10-pixel radius to reduce redundancy, and converted back to world coordinates. Points within 1.0 meter of the agent are excluded to avoid overly local actions. The remaining candidates, expressed in relative polar coordinates (distance and direction), constitute the final structured waypoint set.

Task-Alignment Regulator. After each action, the agent evaluates whether its behavior has advanced toward the current subgoal by comparing visual observations before and after the movement. Specifically, we select a single RGB frame aligned with the selected waypoint direction from the pre-action panoramic observation O_t , and the front-facing RGB frame from the post-action observation O_{t+1} . These two views are concatenated side-by-side into a single image, forming a minimal visual trajectory context for comparison. This composite image, along with the current subtask parsed from the instruction (e.g., “go into the bedroom”), is fed into the VLM. The model generates a natural language description indicating the perceived semantic change. This feedback is then passed to the LLM planner to guide the next decision step, effectively closing the perception-action loop with dynamic task progress estimation.

LLM-Based Planning. We use GPT-4o (accessed via API) as the decision-maker. At each step, GPT-4o receives the instruction L , the current decision space \mathcal{A}_t , and the feedback f_t to decide on the next waypoint $w_t^* \in W_t$. After making a decision, the LLM also generates a short explanation of its reasoning, such as why a particular direction is selected, and maintains a concise history of previous actions and their corresponding explanations.

Stopping Strategy. STRIDER does not rely on the model generating an explicit STOP signal. Instead, the instruction is first decomposed into a series of subtasks by the LLM, with the maximum number of execution steps set to match the number of subtasks, ensuring a hard minimum of 6 steps to guarantee task completion. Once the maximum step is reached, the system halts. This approach eliminates reliance on potentially unreliable stop predictions from the model and guarantees stable behavior across tasks.

Hardware. We conduct all inference experiments on a single NVIDIA RTX 4090 GPU. Since both the VLM and the LLM are accessed through APIs rather than deployed locally, and the original trained waypoint predictor is discarded in favor of structured skeleton-based planning, the overall GPU memory footprint of our system is minimal. This makes our pipeline lightweight and feasible to run. For users interested in local deployment of the LLM, we recommend referring to the hardware setup used in the Open-Nav as a guideline.

A.2 Prompting Templates

We design a set of prompt templates for the VLM tailored to two key components: perception and feedback generation. These prompts are crafted to guide the VLM in producing meaningful scene descriptions, localized object information, and action-oriented feedback.

Specifically, our perception prompts aim to extract global understanding from panoramic visual inputs, as well as detailed descriptions from directional candidate views. Meanwhile, our feedback



Figure 1: **Overview of the designed prompt templates.** Panoramic Perception prompts guide the VLM to generate global semantic summaries from 360° panoramic views. Directional Perception prompts provide localized visual descriptions for each candidate waypoint. Task-Alignment Regulator prompts enable the system to assess action execution status by comparing visual observations before and after navigation steps.

prompts are primarily designed to guide the VLM in assessing task completion status and generating textual outputs accordingly. We present the prompt templates in Fig. 1.

B Robustness Analysis

In this section, we evaluate the robustness of our STRIDER under two types of realistic challenges: sensory sparsity and mid-execution perturbations.

Table 1: **Robustness to perceptual degradation.** We reduce the number of input RGB-D views from 12 to 6, increasing the angular interval between observations to simulate degraded visual perception. Evaluation metrics reflect each model’s ability to navigate under reduced sensory input.

Method	Views	TL	NE↓	NDTW↑	OSR↑	SR↑	SPL↑
Open-Nav-GPT4	12	7.68	6.70	45.79	23	19	16.10
	6	8.25	8.07	40.29	19	10	8.58
		↑7.44%	↑20.45%	↓12.02%	↓17.39%	↓47.37%	↓46.73%
STRIDER	12	8.13	6.91	51.87	39	35	30.30
	6	8.26	7.83	47.98	35	23	20.47
		↑1.60%	↑13.33%	↓7.49%	↓10.26%	↓34.29%	↓32.47%

Table 2: **Robustness to perturbation recovery.** Evaluation metrics are measured after injecting a mid-trajectory perturbation that displaces the agent from its planned waypoint, assessing its ability to recover and complete the navigation task.

Method	TL	NE↓	NDTW↑	OSR↑	SR↑	SPL↑
Open-Nav-GPT4	8.95	8.32	39.26	15	13	10.24
STRIDER	8.47	7.52	47.62	31	28	23.88

B.1 Robustness to Perceptual Degradation

To validate the model’s robustness to sparser perceptual inputs, we deliberately reduce the robot’s accessible RGB-D observations from 12 to 6, increasing the angular interval from 30° to 60° . This setting directly challenges the model’s ability to maintain spatial awareness and semantic consistency with less comprehensive visual coverage. We take Open-Nav-GPT4 as a baseline for comparison to assess the relative robustness of different models under the same degradation condition.

As shown in Table 1, both Open-Nav-GPT4 and STRIDER experience significant performance drops under degraded perception, yet the degree of robustness differs markedly between them. For Open-Nav-GPT4, reducing the number of views results in a 20.45% increase in navigation error (NE), along with substantial decreases in success-related metrics: success rate (SR) drops by 47.37%, and success weighted by path length (SPL) drops by 46.73%. This indicates that the model’s path planning and decision confidence are heavily reliant on high-resolution panoramic context. Furthermore, the decrease in normalized dynamic time warping (NDTW) by 12.02% suggests degraded path adherence and reduced semantic trajectory alignment.

In contrast, STRIDER demonstrates relatively greater robustness. While it also suffers from performance degradation, the increases in NE (+13.33%) and decreases in NDTW (−7.49%) and SR (−34.29%) are more moderate. Notably, SPL drops by 32.47%, which, though significant, is still considerably less than the degradation observed in Open-Nav-GPT4. This suggests that STRIDER benefits from either more efficient path reasoning or a less perception-heavy reliance on global context.

Overall, the analysis reveals that while both models are sensitive to perceptual sparsity, STRIDER retains higher performance under degraded input, potentially due to more robust spatial priors or stronger local visual grounding. These findings highlight the importance of designing VLM-based agents that are resilient to variations in sensory input density, especially in real-world deployment scenarios where perceptual completeness cannot always be guaranteed.

B.2 Robustness to Perturbation Recovery

To validate the robustness of STRIDER in recovering from unexpected execution deviations, we design a perturbation recovery experiment that simulates real-world disturbances during navigation. Specifically, at a randomly selected step during each episode, the agent’s current waypoint is perturbed by altering its location with a slight displacement from the original path. This setting aims to evaluate whether the agent can detect the deviation, reorient itself, and still accomplish the navigation goal effectively.

As shown in Table 2, both Open-Nav-GPT4 and STRIDER experience performance degradation due to mid-trajectory perturbations. However, STRIDER demonstrates significantly stronger robustness across all evaluation metrics. In terms of trajectory length (TL), Open-Nav-GPT4 exhibits a more pronounced increase (from 7.68 to 8.95) compared to STRIDER (from 8.13 to 8.47), indicating that STRIDER recovers more efficiently with fewer detours. Similarly, navigation error (NE) increases to 8.32 for Open-Nav-GPT4 but remains notably lower for STRIDER at 7.52, suggesting better spatial correction and path realignment capabilities.

When examining semantic alignment metrics, STRIDER retains a higher normalized dynamic time warping (NDTW) score of 47.62, compared to 39.26 for Open-Nav-GPT4, reflecting its superior ability to return to semantically plausible trajectories after deviation. Success-related metrics further highlight this gap: STRIDER achieves a 28% success rate (SR) post-perturbation, more than double that of Open-Nav-GPT4 (13%). The success weighted by path length (SPL) shows a similar trend, with STRIDER maintaining 23.88 versus Open-Nav-GPT4’s 10.24, confirming that not only does STRIDER recover better, but it also does so more efficiently.

Overall, these results underscore the importance of robust recovery mechanisms in navigation agents. While both models struggle under unexpected disruptions, STRIDER’s relatively graceful degradation suggests that it possesses stronger global scene grounding and re-planning capabilities. This robustness is critical for real-world deployment, where perfect plan execution cannot be assumed, and systems must dynamically adapt to noisy or unstable environments.

C Visualization Results

To complement the quantitative evaluation, we present a series of qualitative visualizations that offer deeper insight into the agent’s behavior, perception, and reasoning process. These include a detailed case study, additional trajectory examples on benchmark datasets, and representative failure cases.

C.1 Case Study

To better understand how STRIDER perceives its environment, interprets instructions, and makes decisions throughout a navigation episode, we conduct a detailed case study with full-step visualization. As shown in Fig. 5–9, we select a representative trajectory and visualize each decision step, from $t = 1$ to $t = 5$. At each step, we include the panoramic scene description, directional candidate views, the agent’s reasoning process, selected action, and the task-alignment feedback generated after execution.

This case illustrates how the agent sequentially decomposes the instruction, grounds it to visual observations, and dynamically reasons over available waypoints. The visualizations provide insights into the agent’s multimodal understanding, as well as its ability to monitor task progress and adjust behavior based on feedback. This step-by-step breakdown offers a qualitative complement to the quantitative results, highlighting the interpretability of our STRIDER.

C.2 More Visualization Trajectories

To further illustrate the agent’s navigation behavior, we present additional qualitative trajectory visualizations on both the R2R-CE and RxR-CE datasets, as shown in Fig. 2 and Fig. 3. For each selected episode, we visualize the agent’s step-by-step observations in temporal order by arranging the visual input images from left to right. This provides a clear view of how the agent perceives and interprets its environment over time. In addition, we include top-down map visualizations showing both the agent’s executed trajectory (Blue Line) and the annotated reference path (Green Line). The starting position and target location are marked with blue and red dots, respectively. These visualizations provide insights into the agent’s decision-making dynamics, spatial understanding, and alignment with human demonstration trajectories.

C.3 Failure Cases

Despite the overall effectiveness of STRIDER, we observe certain failure cases that reveal limitations imposed by the perception setting, as shown in Fig. 4. The agent is instructed to descend a staircase and proceed toward a designated door. During navigation, the agent only receives horizontally



Figure 2: **Qualitative trajectory visualizations on the R2R-CE dataset.** Each row represents an instruction-following episode, where images from left to right correspond to the agent’s visual observations over time. The green line denotes the reference path, the blue dot indicates the agent’s starting position, and the red dot marks the target location.

aligned RGB-D inputs and cannot perceive content above or below its eye level according to the perception setting. As a result, the downward staircase is not captured in its visual input, leading to a misinterpretation of the scene and incorrect action selection. Additionally, the VLM fails to recognize the “black door” referenced in the instruction during the early stages of execution. This semantic grounding error contributes further to the deviation from the intended trajectory. Such cases highlight the importance of vertical field-of-view coverage and more accurate visual grounding when operating in spatially complex environments.

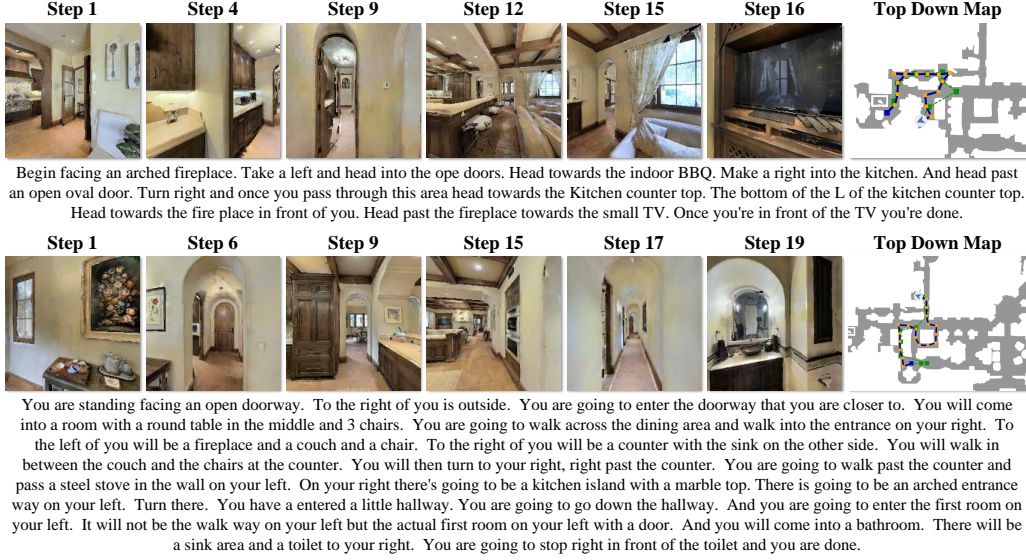


Figure 3: **Qualitative trajectory visualizations on the RxR-CE dataset.** Each row represents an instruction-following episode, where images from left to right correspond to the agent’s visual observations over time. The green line denotes the reference path, the blue dot indicates the agent’s starting position, and the red dot marks the target location.

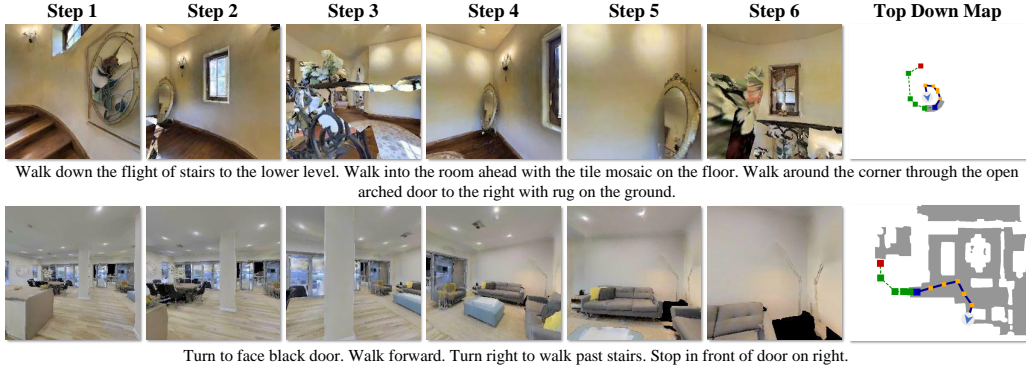


Figure 4: **Failure case caused by limited visual coverage and VLM misperception.** The agent only receives horizontally aligned images and is unable to perceive the staircase below. In addition, the VLM fails to detect the black door referenced in the instruction, leading to incorrect navigation.

D Limitations and Future Work

While STRIDER demonstrates promising performance, there remain several limitations associated with the system setting and underlying model capabilities. First, due to the perception setup, the agent only receives RGB-D observations captured at a fixed horizontal viewing angle. This configuration excludes upward or downward tilt, leading to incomplete vertical field-of-view coverage. In environments with significant elevation changes, such as staircases or split-level layouts, this limitation may result in missing important visual cues that are outside the visible range. Second, our system relies heavily on the capabilities of large-scale VLMs and LLMs. While these models offer powerful generalization across domains, they also exhibit inherent weaknesses. In particular, VLMs may fail to consistently identify task-relevant objects or overlook subtle visual cues due to their goal-agnostic training paradigm. Similarly, LLMs may occasionally generate plausible yet incorrect decisions when provided with ambiguous or incomplete visual descriptions. Since our

method builds on top of these pretrained models, its effectiveness is inherently constrained by their perceptual and reasoning limitations. Lastly, the use of large-scale foundation models introduces additional computational latency. The combined inference time of VLM and LLM modules may limit the system’s applicability in real-time or high-frequency decision-making scenarios. In future work, we aim to explore more perception-rich configurations that include vertical view coverage and investigate more lightweight, powerful VLM variants to improve both perceptual relevance and runtime efficiency.

E Broader Impacts

Our work explores the integration of large-scale vision-language models (VLMs) and language models (LLMs) for embodied navigation, contributing to more generalizable, interpretable, and instruction-following agents in complex 3D environments. This approach has the potential to enhance human-robot interaction in practical scenarios such as indoor service robotics, assistive navigation, and autonomous exploration. By leveraging multimodal understanding and reasoning, our method brings AI agents closer to natural, language-driven task execution. However, the reliance on large pre-trained models also introduces concerns regarding bias, reliability, and computational accessibility. VLMs may inadvertently encode cultural or dataset-specific biases, which could affect perception or decision-making in sensitive environments. Moreover, the high inference cost may limit deployment in resource-constrained or latency-sensitive settings. We encourage future research to investigate fairer, more efficient, and human-aligned multimodal systems, especially in safety-critical applications.

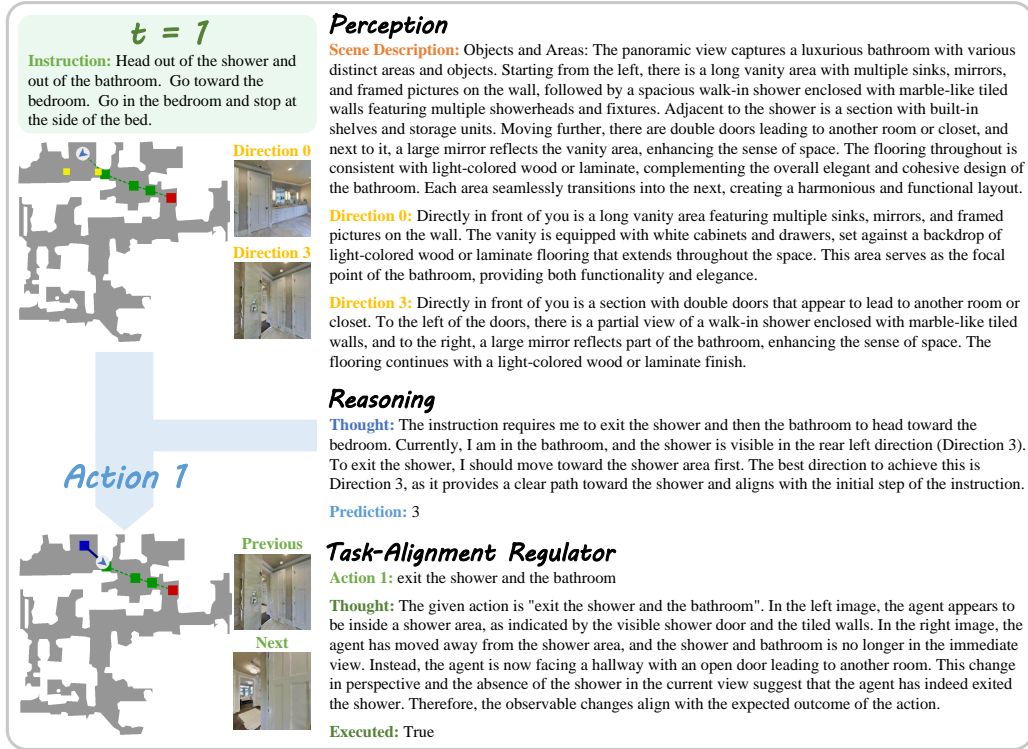


Figure 5: Case study with the navigation process at step 1.

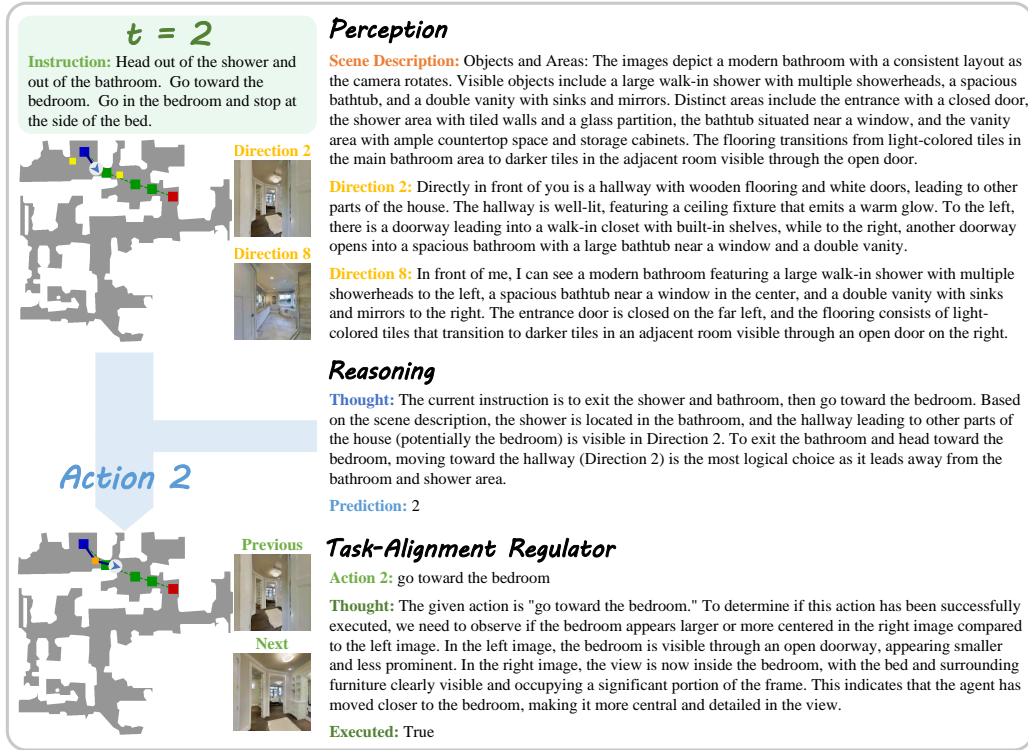


Figure 6: Case study with the navigation process at step 2.



Figure 7: Case study with the navigation process at step 3.

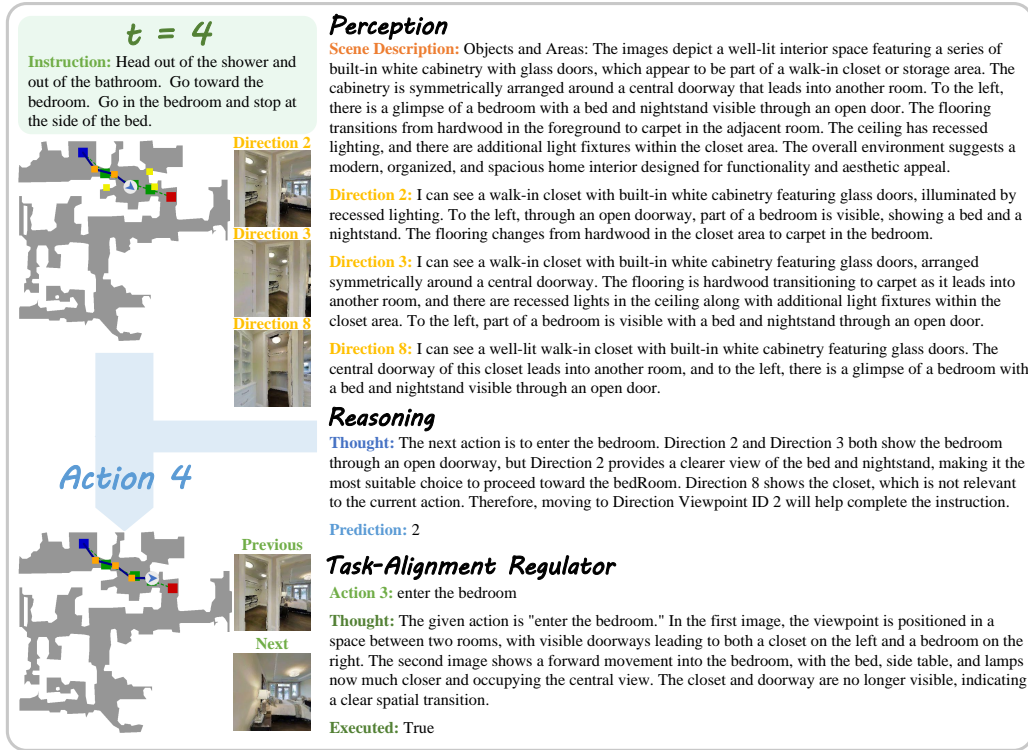


Figure 8: Case study with the navigation process at step 4.

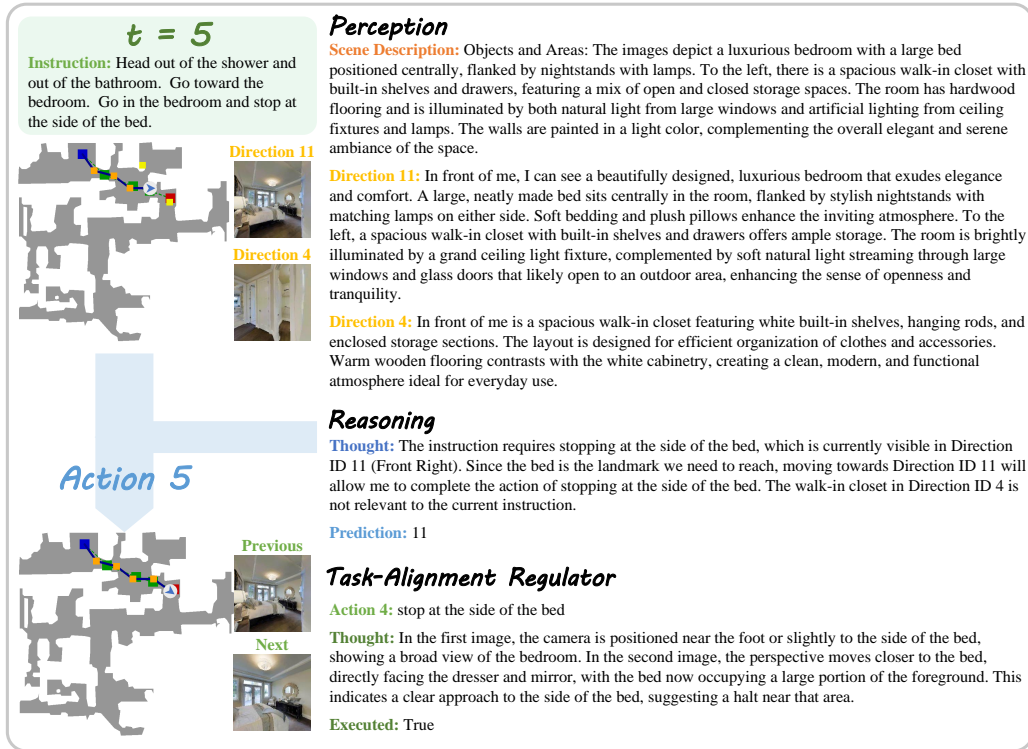


Figure 9: Case study with the navigation process at step 5.