Appendix

A. Implementation Details

Since the most commonly used mobile robots are equipped with ego-centric RGB-D pinhole cameras currently, we primarily evaluate VLN methods without panoramic views in this work. For classic end-to-end single-step discrete action prediction models (Seq2Seq [25], CMA [25], and NaVid [64]), we directly use their publicly available code and pre-trained weights. For the other two model types—the end-to-end continuous multi-step prediction model (RDP) and the map-based LLM model (VLMaps [21])—we introduce several modifications, which are detailed in this appendix.

A.1. Recurrent Diffusion Policy for VLN

The RDP model takes ego-centric RGB-D observations and language instructions as inputs. Since some instructions exceed the 77-word limit in standard CLIP [43], Long-CLIP [63] is used as both the RGB and instruction encoders. The depth encoder follows CMA, using ResNet50 pre-trained on point-goal tasks. Each RGB image is represented by five tokens: The first token encodes the global feature, while the remaining four tokens capture semantic information via grid pooling [64]. The flattened depth features are added to the first RGB token, resulting in a fused visual feature dimension of $\mathbb{R}^{5\times h_d}$, where $h_d = 512$. To improve progress awareness, we incorporate previous 4-step actions (PA) and relative coordinates (RC) from the starting point, both represented in $(\Delta x, \Delta y, \Delta yaw)$. The key difference is that PA encodes the last four steps relative to the current position, while RC represents the current position relative to the starting point.

For historical observation encoding, initially, we experimented with a video-based format, similar to NaVid, where stacked images provided long-term sequence information. However, this approach led to rapid convergence of the diffusion process to small losses, causing severe overfitting. Through our experimentation, we found that employing a recurrent GRU structure to maintain and update historical observations improved generalization:

$$h_t = \text{GRU}([V_c, RC, PA], h_{t-1}). \tag{10}$$

Then, we apply two cross-attention mechanisms to align attended vision ($q = \text{Concat}(h_t, V_c)$) and language features $I = \{w_i\}_{i=1}^{L}$, where each modality serves as the key and value for the other:

$$g_1 = \text{CrossAttn}(q, I, I), \quad g_2 = \text{CrossAttn}(I, q, q). \tag{11}$$

Finally, the condition feature for the diffusion model is formed by concatenating all extracted features:

$$c_t = \text{Concat}(g_1, g_2, h_t, RC, PA). \tag{12}$$

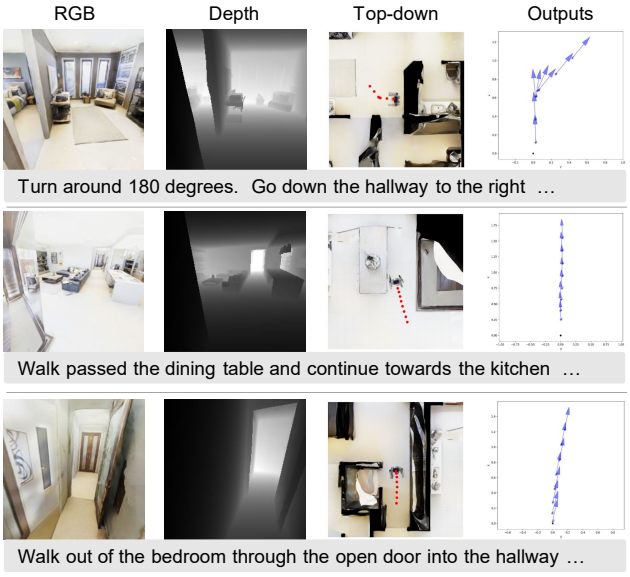


Figure 7. Examples of the robot observations and RDP outputs.

We employ a transformer-based diffusion module [48] with one encoder layer and three decoder layers. During training, the ground-truth trajectory coordinates $T \times (\Delta x, \Delta y, \Delta yaw)$ are perturbed with random noise, and the network is trained to predict and remove this noise. The iterative denoising process follows DDPM [18]. Additionally, we introduce a self-attention-based stop prediction head to determine the current stop progress (from 0 to 1). The stop signal is triggered if: All predicted actions from the diffusion head are below the threshold 0.1, or the stop progress output exceeds 0.8. The output of the RDP is shown in Fig. 7. During navigation, RDP predicts 8 future trajectory waypoints and executes 4 steps per iteration.

In our experiments, RDP demonstrated improvements over the previous baseline models (Seq2Seq and CMA) when trained from scratch. However, there remains significant potential for further enhancement. As this paper primarily focuses on the new physical VLN platform (VLN-PE), we introduce RDP as a baseline method for predicting trajectory waypoints, which can be further integrated with control-theoretic approaches like the Model Predictive Control (MPC) framework to enhance motion smoothness, addressing the jerky transitions seen in discrete action-based methods. We hope this work can inspire and support some future research in this direction.

A.2. Improved VLMaps

VLMaps differs from traditional end-to-end models by utilizing a spatial semantic map representation that directly integrates pre-trained vision-language features of the physical world. This approach enables natural language-based map indexing without requiring additional labeled data. Therefore, we chose this method as one of the technical pipelines

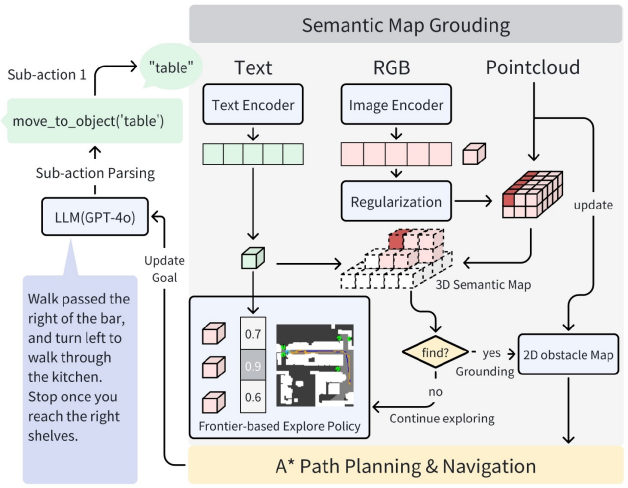


Figure 8. Framework of the improved VLMaps.

for evaluation. However, the original VLMaps lacks a direct exploration policy and struggles with room-level descriptions (*e.g.*, "enter the living room"), which require an agent-oriented perspective rather than reliance on a global semantic map. To address these limitations, we improved the VLMaps framework (as shown in Fig. 8) with two key enhancements.

Exploration Policy: Inspired by VLFM [62], we implement a frontier detection strategy, where a frontier is defined as the boundary between explored and unexplored areas. When the robot fails to detect the target object from the current pixel embeddings, it performs a turn-around maneuver, moves to each frontier, and evaluates whether the viewpoint is likely to contain the next landmark using the image and text encoders from LSeg. For instance, we observed that "table" exhibits a higher similarity score with scenes of "dining room" compared to "toilet," validating the policy's ability to guide the robot toward plausible directions.

Room-Level Descriptions: Similarly, we leverage the CLIP module from LSeg as a classifier to assess whether the viewpoint aligns with the intended room context. Specifically, we use a predefined set of room names ("living room," "dining room," "bedroom," "kitchen," "toilet," "others") as text inputs to index the current RGB image. Upon successful room detection, we naturally incorporate actions such as `self.move_to_room('room_name')`.

In the VLN-PE, we apply additional techniques to reduce the fall rate and stuck rate. For example, we implement an A* algorithm as the local planner, assigning higher costs to dilated and unexplored areas. When executing commands like `self.move_forward(1)`, the robot may collide with obstacles if not properly oriented. To address, we define a cost function to identify the optimal node n^* from the robot's perspective: $n^* = \arg\min_n(||dist(n, x_0) - dist(x_g, x_0)|| + \alpha\gamma)$, where x_g is the goal position, x_0 is the current position, $\alpha = 0.25$ is a weight parameter, and γ is the angle required to face the target. This ensures the robot slightly reorients itself before moving forward, minimizing collision risks. An example of the improved VLMaps is shown in Fig. 9. Compared to end-to-end methods, map-based modular approaches offer more explainable and reliable results. However, their performance heavily depends on mapping and localization accuracy, which could limit the practical deployment.

A.3. Experimental Details

All training experiments are conducted using NVIDIA RTX 4090 GPUs. The CMA and Seq2Seq models are trained on a single GPU with a batch size of 2, requiring approximately one day to converge. The RDP model is trained on 4 GPUs using PyTorch's DataParallel module, with a total batch size of 8, and completes training in around two days. All models are optimized using the AdamW optimizer with a learning rate of 1×10^{-4}. The maximum trajectory length is set to 200. For evaluation, the CMA model requires approximately 4 hours to complete a full evaluation on the R2R-CE benchmark when run in parallel on 8 GPUs.

A.4. Datasets

The trajectories sampling strategy for the newly introduced datasets (GRU-VLN10 and 3DGS-Lab-VLN) is as follows: (a) generate a freemap, (b) randomly sample start-goal pairs, and (c) filter out invalid paths (overly short, long, or similar ones). Instructions are generated via a modular pipeline [17] with action and environment recognition, GPT-4 in-context description, and human refinement. Comparisons of datasets are presented in Fig. 10.

A.5. Metrics

Metrics. Following standard VLN evaluation protocols [2, 25], we use five primary metrics: *Trajectory Length (TL)*, measured in meters; *Navigation Error (NE)*, which quantifies the distance between the predicted and actual stop locations; *Success Rate (SR)*, indicating how often the predicted stop location falls within a predefined distance of the true location; *Oracle Success Rate (OS)*, which assesses the frequency with which any point along the predicted path is within a certain distance of the goal; and *Success Rate weighted by Inverse Path Length (SPL)*, which balances success rate with path efficiency. As physical realism is a key focus of this work, we introduce two more metrics: *Fall Rate (FR)*, which measures the frequency of unintended falls, and *Stuck Rate (StR)*, which quantifies instances where the agent becomes immobilized. Specifically, *"Fall"* is the robot having a roll $> 15°$ or pitch $> 35°$, or a center-of-mass–to-foot height below a robot-specific threshold. *"Stuck"* is defined as both position and heading change < 0.2m and $15°$ for 50 steps.

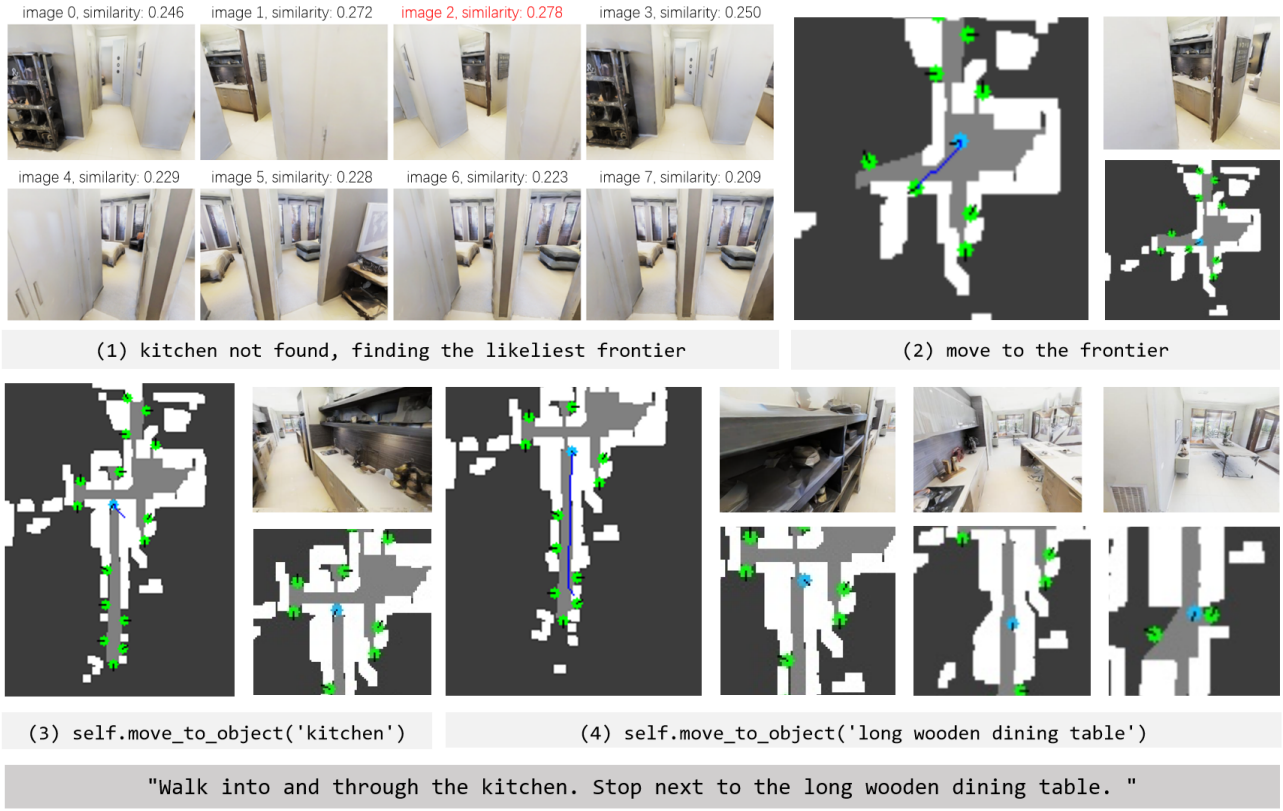


Figure 9. Example of improved VLMaps. Blue dot: current position (black line: orientation). Green dot: frontiers (black line: exploration orientation). White: dilated obstacles. Light gray: explored area. Dark gray: unexplored area. Blue line: local planner trajectory.

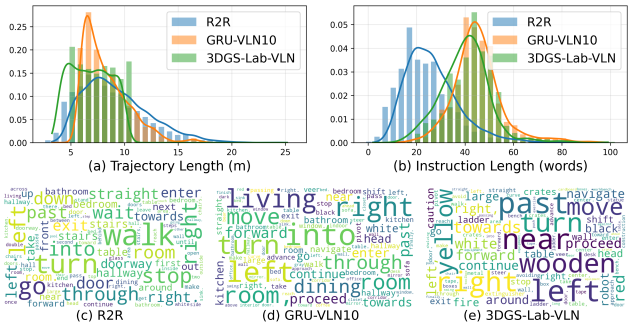


Figure 10. Comparison of distributions across datasets.

A.6. Controllers

Thanks to NVIDIA Isaac Sim's advanced physical simulation capabilities, we can seamlessly apply various control theories, making the low-level control policy more diverse and aligned with real-world robotic applications. In this work, we utilize three types of controllers for experimentation: flash control, move-by-speed control, and move-along-path control.

- Flash Control: This mechanism mimics platforms that lack physical cross-embodiment support, allowing the agent to instantly reach the target position without considering physical motion constraints.
- Move-by-Speed Control: This method simulates realistic motion dynamics by controlling the agent's velocity using linear and angular speed commands. For legged humanoid and quadruped robots, we employ the RL-based policies to regulate movement, ensuring the robot follows the required forward and rotational speeds. For wheeled robots, we use a differential drive controller to manage navigation. For end-to-end models, we implement discrete actions using this controller.
- Move-along-Path Control: This approach enables the agent to follow a predefined trajectory, replicating path-following behaviors in robotic navigation. For the Map-based method (VLMaps), we apply the A* path planning algorithm and use this controller with a PID system to ensure smooth trajectory following.

A.7. Fine-tune on the specific datasets

To better evaluate out-of-MP3D-style domain generalization, we collect additional VLN datasets using GRUScenes and 3DGS-rendered environments. Since these datasets are primarily used for evaluation, only a small portion of the

data is allocated for training, while the majority is reserved for testing. For CMA and RDP, all training experiments use a learning rate of 1e-4 with a cosine learning schedule. In Tab. 2 and Tab. 3, "w/o FT" refers to direct zero-shot transfer using VLN-PE-R2R-trained weights for evaluation without further in-domain fine-tuning in these new scenes. On the GRU-VLN10 dataset, small models significantly improved after 10 epochs of fine-tuning, whereas the SoTA large model NaVid showed limited zero-shot performance. This highlights the limited diversity of existing VLN benchmarks, which could not fully assess model generalization.

B. Impact of Sim-to-Real Transfer

Compared to traditional VLN simulators and platforms, VLN-PE introduces a significant advancement by supporting physical VLN across diverse robot types, enabling data collection, training, and closed-loop evaluation in physical settings. We begin by identifying the limitations of existing VLN algorithms when deployed in physical environments, as initially verified within a physics-enabled simulator. After fine-tuning on data collected through VLN-PE, we observe consistent performance improvements within the simulated physical setup. Encouraged by these results, we further evaluate our approach in real-world settings. Specifically, we conduct experiments using a Unitree Go2 robot equipped with an Intel RealSense D455 RGB-D camera across 14 indoor episodes (see Table 7 and Fig. 11). The model fine-tuned with VLN-PE demonstrates improved adaptation and generalization, confirming the practical effectiveness of our platform.

In particular, we observe that the CMA baseline model, after VLN-PE fine-tuning, exhibits more confident forward movement and better semantic grounding during navigation. In contrast, the CMA Full baseline trained solely on VLN-CE struggles in real-world conditions, frequently resulting in aimless rotation and poor generalization. One notable remaining challenge is the handling of the stop action.

Method	Fine-tuned on VLN-PE	OS↑	SR↑
CMA	✗	14.29	7.14
	✓	**57.14**	**28.57**

Table 7. Impact of VLN-PE on real-world performance.

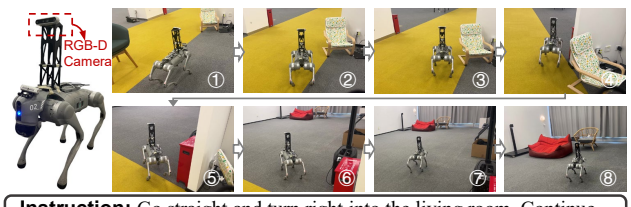


Figure 11. Real-world experiments using a Unitree Go2 robot.

CMA often fails to robustly predict when to stop. To mitigate this, we let the robot output the stop action when the predicted probability of the stop action exceeds 1×10^{-4}.

C. Analysis of Failure Cases

In Tab. 3, we observe that the SoTA ego-centric model, NaViD [64], shows exceptionally poor zero-shot performance (*e.g.,* 5.8 SR and 1.0 SPL) on our 3DGS-Lab-VLN datasets. The possible reasons for the performance degradation could be summarized as follows: Firstly, the use of 3D Gaussian Splatting (3DGS) for rendering may introduce artifacts and distortions. As shown in Fig. 13, rendering artifacts can cause some blurring in ground areas and distant details, introducing subtle distortions that may go unnoticed by the human eye. In our experiments, the model relying solely on RGB input is highly vulnerable to such pixel-level noise, leading to failure in affected scenes. This underscores the need for research on image perturbations and related safety issues in VLN models. Additionally, we note that the NaViD model frequently rotates in circles to find a better viewpoint for localizing the target, which accounts for 70% of failures (Fig. 12). In summary, our findings on the limitations of current SoTA VLN methods align with the conclusions of this paper. We hope our insights and tools will drive the development of more robust and generalizable VLN models, especially in diverse, non-MP3D-style environments.

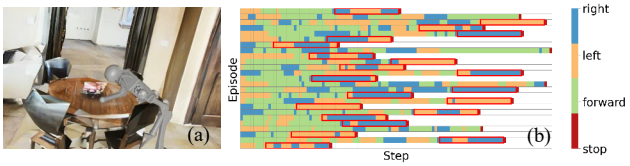
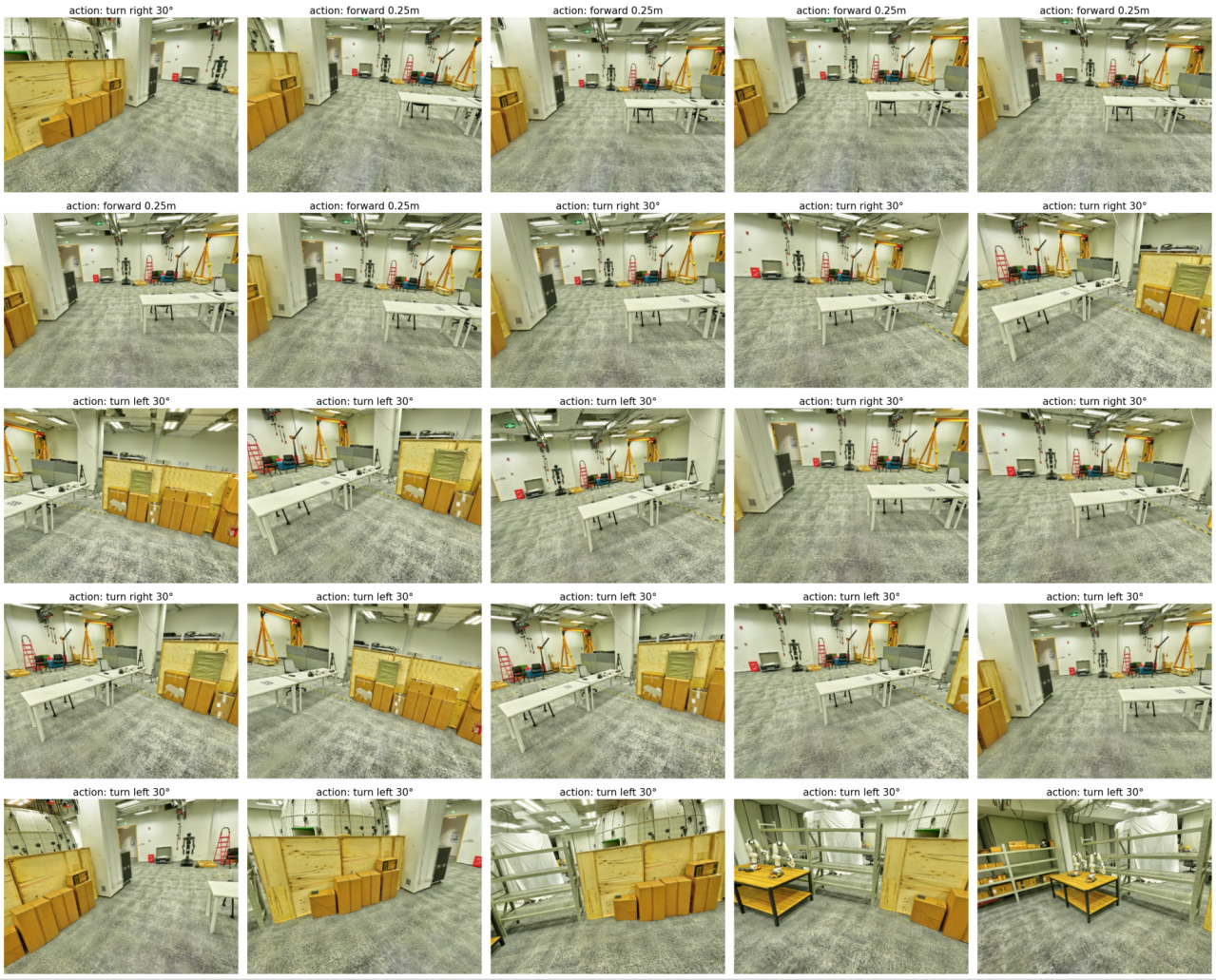


Figure 12. Visualization of the failure cases. (a) shows a typical failure case where the agent collides and falls. (b) highlights NaVid's repetitive turning before stopping (red box).

D. Limitations and Future Work

While this work evaluates various ego-centric VLN methods, some other state-of-the-art VLN approaches rely on panoramic observations [1, 3, 24]. These methods use panoramic views with depth to generate sparse waypoint connections, integrating them with discrete VLN techniques for path selection—an approach that has demonstrated strong performance in previous non-physical settings. Since our primary goal is to evaluate the existing VLN methods under the physical settings, we adopt an ego-centric view setting to align with current robotic perception systems. However, as robotics evolves—potentially resembling autonomous driving systems with more diverse RGB or radar sensors—future robots may benefit from panoramic



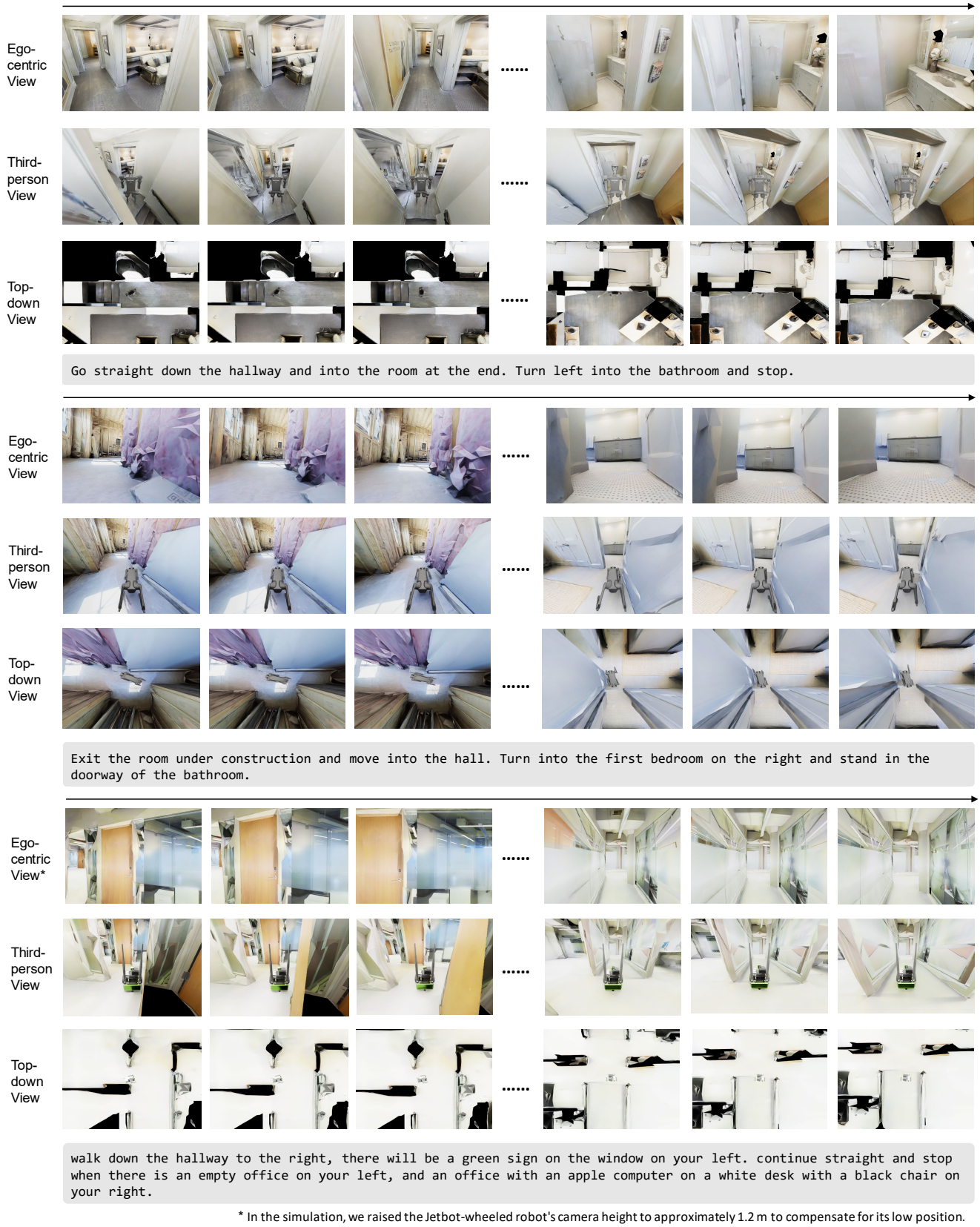
"Steer right towards the red ladder, then arrive at the white table with black chairs. Turn left near the yellow crane, advance past the robot statue, and move forward towards the desk with chairs. Finally, head left towards the office chair."

Figure 13. Visualization of the failure case for the ego-centric SoTA VLN model, NaViD, in 3DGS online-rendered scenes. The model tends to predict rotation actions, indicating a failure to interpret the intended trajectory.

perception. Thus, we plan to extend our evaluation to panoramic VLN methods in future work. Additionally, with multi-robot support and real-time 3DGS scene rendering, our platform has significant potential to facilitate a real-sim-real VLN pipeline, enhancing real-world adaptability for embodied agents in familiar environments. We leave this for future research.

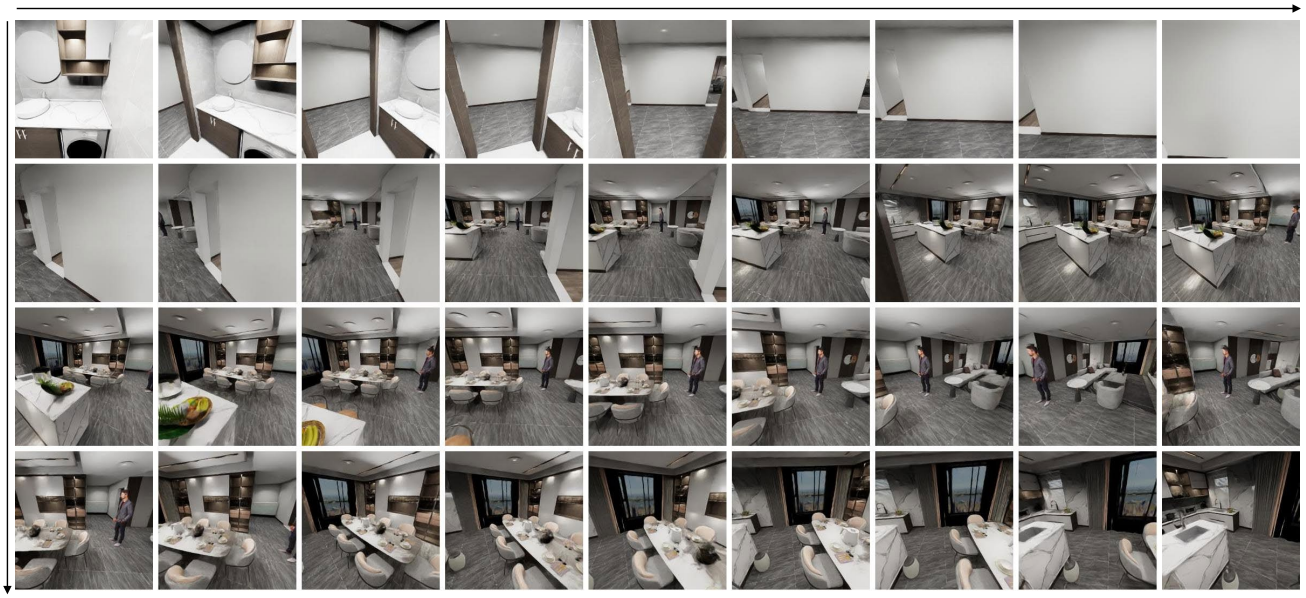
E. Additional Qualitative Examples

To better illustrate the observations and environments within VLN-PE, we provide supplementary videos showcasing the significant shaking and instability experienced by physical agents during navigation. Additionally, Fig. 14 presents different viewpoints—ego-centric, third-person, and top-down—using various robot types in VLN-PE. Fig. 15 displays trajectories and instructions from our newly introduced 10 high-quality synthetic scenes (GRU-VLN10) and a 3DGS online-rendered scene (3DGS-Lab-VLN), supporting out-of-MP3D-style evaluations.

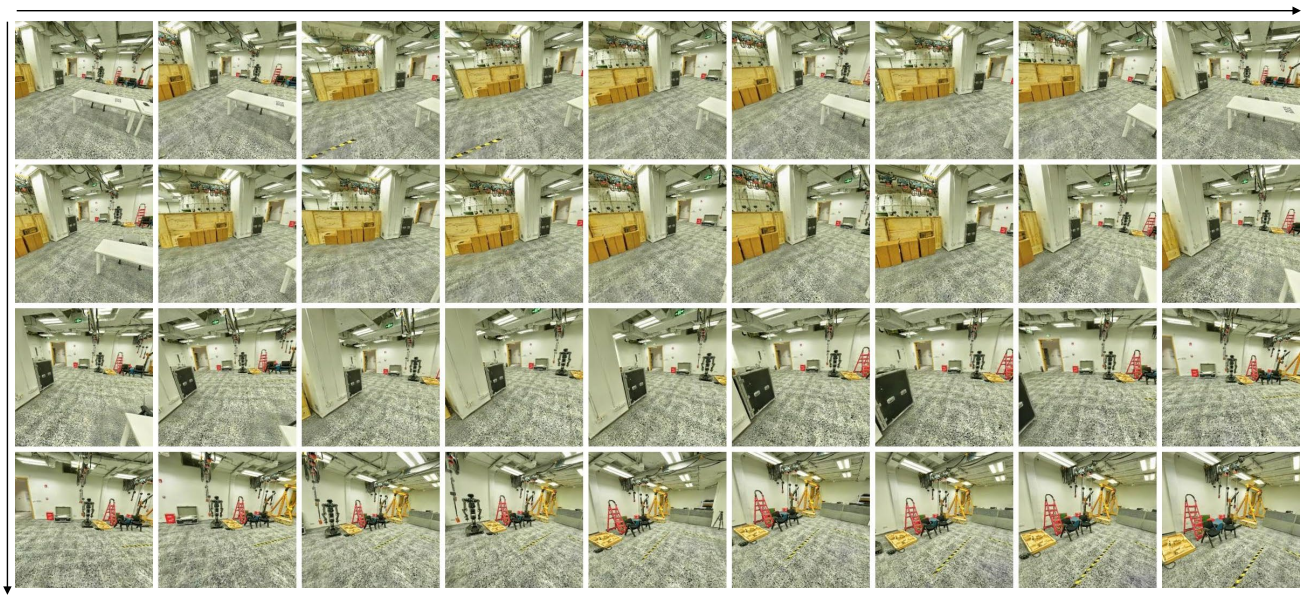


* In the simulation, we raised the Jetbot-wheeled robot's camera height to approximately 1.2 m to compensate for its low position.

Figure 14. Visualization of different robot viewpoints in VLN-PE. Leveraging the powerful interactive capabilities of Isaac Sim, researchers can easily observe robot motion from various perspectives within the environment.



Turn left into the bathroom with a round mirror above the sink. Move through the indoor space, steering left into the empty room. Proceed to the living room, then enter the dining room with a table and chairs near the window. Continue straight to the kitchen, stopping at the sink.



Turn right, move forward past the wooden counter to the long white table, then turn left at the workbench. Continue forward, passing the white bench and wooden crates, then enter near the wall with exposed pipes. Finally, turn left and stop by the ladder.

Figure 15. Examples of trajectories and instructions from our introduced GRU-VLN10 and 3DGS-Lab-VLN datasets.