VL-Explore: Zero-shot Vision-Language Exploration and Target Discovery by Mobile Robots

Yuxuan Zhang[1], Adnan Abdullah[2], Sanjeev J. Koppal[⋆3], and Md Jahidul Islam[⋆4]
[⋆]Equal Contribution [*†‡]

Abstract

Vision-language navigation (VLN) has emerged as a promising paradigm, enabling mobile robots to perform zero-shot inference and execute tasks without specific pre-programming. However, current systems often separate map exploration and path planning, with exploration relying on inefficient algorithms due to limited (partially observed) environmental information. In this paper, we present a novel navigation pipeline named "VL-Explore" for simultaneous exploration and target discovery in unknown environments, leveraging the capabilities of a vision-language model named CLIP. Our approach requires only monocular vision and operates without any prior map or knowledge about the target. For comprehensive evaluations, we designed a functional prototype of a UGV (unmanned ground vehicle) system named "Open Rover", a customized platform for general-purpose VLN tasks. We integrated and deployed the VL-Explore pipeline on Open Rover to evaluate its throughput, obstacle avoidance capability, and trajectory performance across various real-world scenarios. Experimental results demonstrate that VL-Explore consistently outperforms traditional map-traversal algorithms and achieves performance comparable to path-planning methods that depend on prior map and target knowledge. Notably, VL-Explore offers real-time active navigation without requiring pre-captured candidate images or pre-built node graphs, addressing key limitations of existing VLN pipelines.

Keywords. Vision-Language Navigation; Zero-Shot Visual Servoing; Path Planning; GPS-denied Navigation.

1. Introduction

Autonomous robots face considerable challenges in exploring and discovering targets within unknown environments without a prior map, typically requiring a dedicated mapping phase before initiating path planning strategies [8, 12, 23]. Recent vision-language (VL) approaches have begun to address this by incorporating long-term memory architectures, a key factor driving performance gains in navigation tasks [6]. However, methods that rely on pre-constructed maps or pre-existing environmental knowledge often exhibit poor adaptability to dynamic or evolving conditions, necessitating frequent reinitialization or bespoke algorithmic interventions [1, 15]. These limitations are especially pronounced in critical real-world applications—such as search-and-rescue, environmental monitoring, and warehouse exploration—where adaptability and robustness are essential [20, 26, 40]. Despite the importance of efficient, zero-shot exploration in such scenarios, current VLN systems have yet to adequately address this capability.

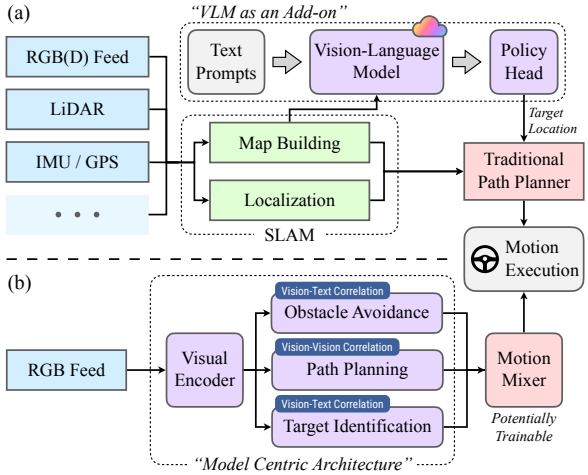


Figure 1. (a) Existing VLN systems generally use VLMs as a complementary add-on; (b) The proposed *VL-Explore* pipeline is designed with a general-purpose VLM at its core, eliminating the need for multimodal sensing and extra processing steps.

When initializing inside an unexplored environment, many traditional methods use a fixed policy that often gets stuck in local minima, leading to inefficient or incomplete exploration [21]. To address this, contemporary VLN systems use high-sensing modalities [6, 12] such as multiple cameras, depth camera or LiDAR to help with first-time exploration. For instance, a 2D LiDAR is commonly used for obstacle avoidance and mapping [35, 38], while a sep-

[*2,4]RoboPI Laboratory, Dept. of ECE, University of Florida (UF)
[†1,3]FOCUS Laboratory, Dept. of ECE, University of Florida (UF)
[‡3]Amazon Robotics (Dr. Koppal holds concurrent appointments as an Associate Professor of ECE-UF and an Amazon Scholar at Amazon Robotics. This project is performed at UF, not associated with Amazon.)

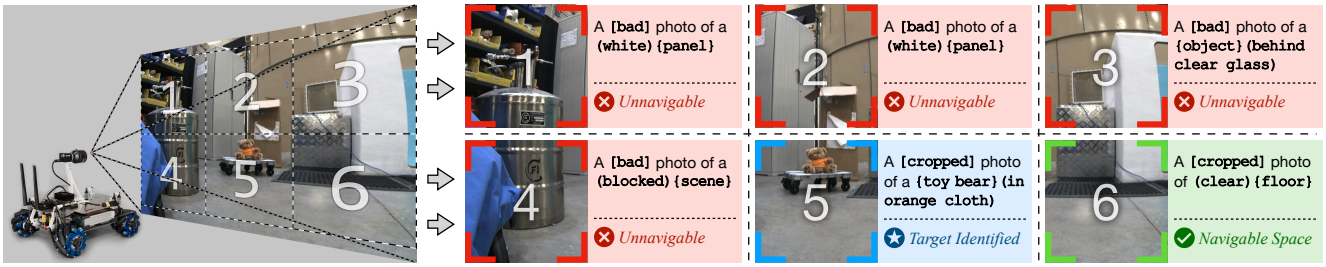


Figure 2. Illustration of an ongoing exploration and target discovery task by the proposed *VL-Explore* system. **Left**: A synthesized image representing the robot's perspective from its onboard camera; the cone illustrates its field-of-view, divided into six numbered tiles for detailed analysis. **Right**: The vision-language perception of each tile is processed by a novel correlation middleware; The motion mixer engine is currently in 'target lock' mode, prioritizing the target (a teddy bear) in tile-5 over other navigable regions (*e.g.*, tile-6) due to the higher precedence assigned to the target.

arate camera is used for scene classification and target detection [22]. Additionally, the initial map traversal is not feasible for partially observable and dynamic environments, requiring frequent re-initialization [5, 16]. To this end, the efficiency and robustness of target discovery by **zero-shot exploration** remain largely under-explored.

In this paper, we present "**VL-Explore**", a novel navigation framework for zero-shot exploration and target discovery by unmanned ground vehicles (UGVs) in unknown environments. VL-Explore extends the spatial context awareness capabilities of general-purpose VLMs [7] to guide 2D robotic exploration and target discovery without requiring access to a prior map. It addresses the limitation that existing VLN systems predominantly use a VLM as a complementary add-on for policy projection, outside the navigation subsystem; see Fig. 1. Instead, ***VL-Explore*** **uses a general-purpose VLM as its navigation core**, with the entire system operating within it. It also eliminates the need for any additional sensing modalities and map building. As illustrated in Fig. 1b, simplifying the system architecture in this way significantly reduces the system complexity and cost, making it more scalable and suitable for resource-constrained systems.

VL-Explore eliminates the contemporary VLM system's **reliance on pre-built knowledge graphs** and candidate image libraries. Specifically, SOTA systems such as ClipNav [9], Clip on Wheels [10], and GCN [23] require pre-captured images of candidate scenes, target locations, and/or pre-built knowledge graphs of the navigable space as prior [8]. Due to this complexity, they are generally tested in 2D simulations or overly simplistic environments. Other contemporary methods, such as SEEK [12], require high-sensing modalities involving multiple cameras, LiDAR, and onboard SLAM pipelines for navigation. In *VL-Explore*, we formulate the problem from a different perspective. Instead of relying on a separate traditional mapping and path-planning backbone, our proposed architecture uses the visual embeddings encoded by a foundational VLM as the **only input to the system**. High-level functionalities such as obstacle avoidance, path planning, and target identification are built upon this foundation for active visual perception.

To validate the proposed framework in real-world environments, we develop a UGV platform that meets the computational demands of VLMs while offering superior maneuverability and mechanical stability during the task. We deployed the CLIP model [18, 36] in our proposed architecture and performed extensive tests in a real-world environment. The results demonstrate that VL-Explore consistently outperforms map-traversal algorithms and achieves performance comparable to path-finding methods, despite the latter relying on prior knowledge of the map and target location. Moreover, VL-Explore achieves significantly shorter trajectory lengths, making it more efficient than map-traversal algorithms. In comparison with state-of-the-art systems, VL-Explore achieves comparable or even superior performance to systems with higher sensor modalities or even with long-term memory. Notably, VL-Explore offers such performance margins without additional sensor modalities, also requiring no separate dedicated map-building or localization modules.

Overall, we make the following contributions in this paper:

1. We propose a model-centric navigation pipeline "**VL-Explore**" for simultaneous exploration and target discovery by UGVs in unknown environments. It requires no additional sensor modalities other than RGB feed for zero-shot active navigation.
2. We design a novel evaluation metric "**Entropy Preserving Score**" (EPS) to measure the exploration efficiency, improving the consistency of score across different experimental environments hence providing better comparability across different systems.
3. We perform extensive experiments with the proposed system and compare its performance against other systems, demonstrating the effectiveness of the proposed system in real-world environments.

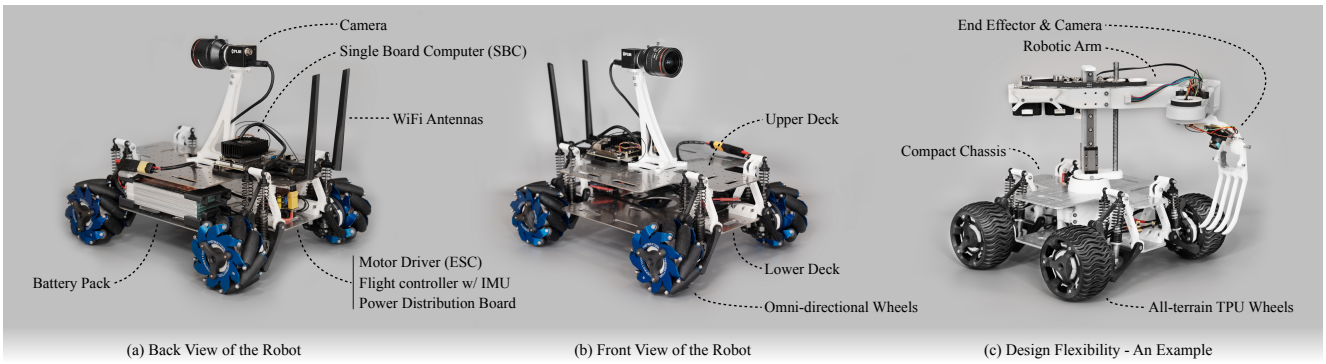


Figure 3. Our *Open Rover* platform designed for *VL-Explore* is shown: (**a**) Back view, showing the single board computer (SBC) and electronics stack including a brushless motor speed controller, a flight controller with IMU, and a power distribution board; (**b**) Front view, showing the camera for zero-shot navigation; (**c**) An alternative design, demonstrating the flexibility of this platform, configured with revised wheels, chassis, and additional manipulators for potential field robotics applications.

2. Related Work: Zero-shot Learning and Vision-Language Navigation

Zero-shot learning enables autonomous robots to identify unseen objects and make navigation decisions in unfamiliar environments [14]. Researchers use visual features [41], knowledge graphs [33], semantic embeddings [43], and spatial appearance attributes [30] to encode information of object classes into the search space. More recently, text-based descriptions or human instructions are combined with vision to improve servoing, as demonstrated in LM-Nav [37] and InstructNav [27]. Among CLIP [36]-based frameworks, CoW [11] presents a trajectory planning strategy using frontier-based exploration. ClipNav [9] designs a *costmap* to further improve obstacle avoidance during exploration. Other approaches such as ESC [45] and VLMaps [17] use prompts to translate action commands into a sequence of open-vocabulary navigation tasks for planning. Besides, CorNav [25] includes environmental feedback in the zero-shot learning process to dynamically adjust navigation decisions on the fly.

Contemporary works on vision-language navigation (VLN) focus on enabling robots to understand language instructions for interactive task planning. VLN algorithms integrate visual information (*e.g.*, recognizing objects, obstacles) and language instructions to plan a trajectory or action sequence for task execution. This is achieved by first generating semantic tags into the SLAM pipeline [10, 14, 17] to generate a 3D map. Subsequently, the resulting augmented map is used to support online navigation to perform zero-shot semantic tasks [42]. These works primarily focus on adding semantic information to existing mapping and exploration techniques [17, 24], but do not participate in the path planning process.

In contrast, *node graph* based VLNs [4, 8, 23] construct a graph of the active map where each node represents a free location and edges stand for directly navigable paths. Dynamic expansion of a node graph is then achieved with the help of precise localization, typically achieved by additional sensors. Experiments show that these algorithms suffer from miscorrelating existing nodes when revisiting an explored area, and also in dynamically changing scenes. To address this, researchers are exploring zero-shot learning aided observation of unknown spaces [42]. Techniques like odometry, stereo depth, and LiDAR sensors are used alongside a pretrained vision-language model to achieve simultaneous mapping and exploration – demonstrating a clear advantage regarding the efficiency of reaching the prompted goal.

The existing zero-shot learning and VLN approaches rely on dedicated sensors to construct a global map on the first run. Subsequent motion decisions are generated by a traditional trajectory planner. Online planning and navigation in dynamic and unknown spaces without a prior map is still an open problem, which we attempt to address in this paper.

3. *Open Rover*: Platform Design

We develop a novel robotic platform "*Open Rover*". The platform is designed to be flexible with drive types, chassis sizes, sensor and actuation add-ons, so it can adapt to the needs of prospective research. The major sensors and actuation components are shown in Fig. 3; it includes a monocular RGB camera and a 2D LiDAR for exteroceptive perception. Four independent wheel assemblies are responsible for actuation; each wheel assembly is modular and self-contained, *i.e.*, it consists of a gearbox, a brushless DC motor, and a suspension system; see Fig. 4b. It also includes a pseudo-odometer that uses telemetry data from an electronic speed controller (ESC), which drives the motors. Additionally, the driver stack consists of a flight-controller module originally designed for quadcopters that reports on-

board sensory data to the host computer for planning and navigation.

Unlike existing platforms like the *TurtleBots* [2], *Open Rover*is designed with an open and spacious chassis to facilitate easy adaptation to any single board computer (SBC). In our setup, we configured the platform to: (**i**) deliver sufficient computational power to handle a general-purpose vision-language model and other computationally intensive tasks; (**ii**) have enough mechanical stability to hold the camera in an elevated position without excess vibrations even on uneven surfaces; and (**iii**) support 3-DOF motions (forward/backward, sideways and rotation), whereas most existing UGV systems have only 2-DOF: forward surge and twist rotation.

3.1. Task Specific Design for VL-Explore

We customize the *Open Rover* platform with a configuration that supports real-time VLN capability for 2D environments. An omnidirectional drive system is chosen to account for the moderate surface irregularities. The four wheels are connected to separate brushless DC motors via a planetary gearbox with a reduction ratio of 16:1. This allows the robot to travel at higher speeds and easily overcome moderate obstacles. A throttle limit of 20% is imposed to ensure operational safety, capping the robot's maximum speed at approximately 2 meters per second.

As shown in Fig. 4b, the gearbox and brushless DC motor are integrated into the wheel hub, optimizing the spatial efficiency and smoothness of the drive system. Moreover, the computing pipeline is powered by an Nvidia `Jetson Orin` module. A `FLIR` global shutter RGB camera, lifted approximately 30 cm above ground, is used for visual perception. This design ensures that the optical center of the lens aligns with the geometric center of the robot. Besides, the *heading* value from the flight controller's IMU facilitates 360° scanning capabilities. A 2D LiDAR is mounted at the front of the upper deck as is shown in Fig. 3b. The LiDAR is only used for safety and visualization purposes and does not influence VLN's decision-making. Further details regarding the LiDAR's utility are explained in Sec.4 of the supplementary materials.

3.2. Features and Capabilities

One unique advantage of our *Open Rover* system design is the compactness of its wheels and motor assembly. The four-wheel independent suspension extends its application to uneven surfaces; when using the *Discovery Wheel* (TPU) shown in Fig. 3c and Fig. 4b, it performs well on grasslands, rocky pavements, and even sand (with reduced maneuverability). The suspension system can also be configured to filter out surface bumps and provide better stability for onboard sensors.

The *Open Rover* platform can be configured to switch between different *drive types* (omnidirectional or differential), *chassis sizes* (with parameterized CAD design), and task-specific *actuation and sensory add-ons* (cameras, LiDARs, manipulators for both indoor and field robotics applications). The chassis plates are designed to house various additional sensors and actuators according to tasks. A comparison of *Open Rover* with two widely used *TurtleBot* UGV variants is presented in Table 1.

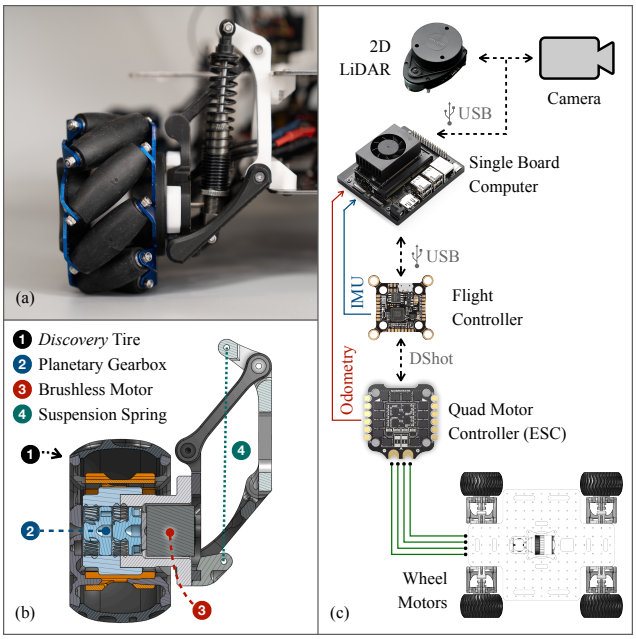


Figure 4. The wheel hub and suspension design are shown on the left: (**a**) suspension system coupled with the omnidirectional wheel; and (**b**) cross-section view of the 'discovery wheel', in-wheel drive system, and suspension links, assembled and rendered by CAD software. The connection diagram is shown on the right: (**c**) the dashed line represents physical connection, while solid lines represent logic functions.

Table 1. Comparison of *Rover Master* with other standard UGV platforms is shown; the acronyms: **[DF]** Differential, **[OD]** Omni-directional, **[CFG]** Configurable.

Platform	Drive Type	SBC	Onboard Sensors	Add-ons	Est. Cost (USD)
TurtleBot3 (Waffle Pi)	**[DF]**	RPi4	PiCamera, IMU, LDS laser	Limited	1500
TurtleBot4 (Standard)	**[DF]**	RPi4	Stereo camera, IMU, RPLiDAR	Limited	2100
Rover Master	**[DF]** / **[OD]**	**[CFG]**	RGB camera, IMU, LiDAR	**[CFG]**	**650** / 850

4. VL-Explore: Navigation Pipeline

Recent advancements in VLMs have demonstrated their potential for spatial reasoning and awareness capabilities [7, 9]. Different from traditional approaches, this study explores the feasibility of incorporating VLMs directly into a

robot's autonomy pipeline to facilitate real-time exploration and target identification. To this end, we introduce a modular pipeline that integrates stages for perception, planning, and navigation. The core computational components of the proposed VL-Explore navigation pipeline are depicted in Fig. 5.

The proposed pipeline comprises three main stages. The **frontend** processes raw input frames, divides them into tiles, and encodes these tiles into embeddings—numerical vectors representing semantic meanings; In our implimentation, the CLIP vision encoder is employed as the frontend. Then, **middlewares** take the visual embeddings generated by the frontend as input and produce scores that carry specific semantic interpretations. These scores are typically derived by correlating the input embeddings with the middleware's internal database. The meanings of these scores can vary depending on the application's requirements. In this study, the middleware generates three types of scores: *navigability*, *familiarity*, and *target confidence*.

Lastly, at the **backend**, those scores from middlewares are used to make motion decisions. It is designed to be adaptable, allowing the integration of different algorithms tailored to various applications and environments. For this study, a minimal backend was implemented with three operational modes: basic navigation, look-around, and target lock. These modes are dynamically activated or deactivated based on the provided scores.

Notably, the proposed VL-Explore pipeline is *agnostic to* CLIP or any other VL backbone. OpenCLIP is used in our experiments given its open-source status and 3-4 years of maturity in various research fields, ensuring reproducibility. While a more recent multi-modal VLM could potentially enhance the performance of the suggested pipeline, the primary focus of this work is to validate the feasibility of integrating VLMs into a real-time navigation system. Thus, we chose to utilize a well-established model for better reproducibility.

4.1. Visual Perception Frontend

In the frontend, raw camera frames are sliced into six tiles; the slicing strategies are discussed in Sec.1 of the supplementary materials. Each tile represents a spatial location in the robot's FOV, which are scaled and processed by CLIP's visual encoder. As shown in Fig. 5 f, each frame is sliced into $N = 6$ tiles and then rearranged into a tensor of shape $N \times 3 \times H \times W$, with H and W being the tile height and width in pixels. The encoder processes these inputs and generates an $N \times D$ embedding vector for each tile, where D is the dimensionality of each prediction vector ($D = 512$ in the CLIP model).

Additionally, the standard deviation for each tile is computed and combined with the model's predictions. This metric serves as an indicator of the *amount of information*

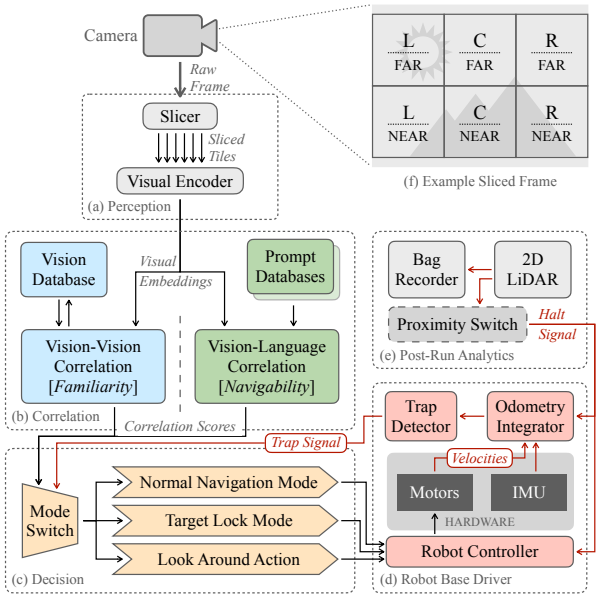


Figure 5. An overview of the architecture of VL-Explore; the *primary* data flow and *supplementary* sensory feedback are marked by black and red arrows, respectively. The camera frame-slicing strategy is shown in (**f**); each frame is sliced into two rows (NEAR, FAR) and three columns (LEFT, CENTER, and RIGHT). Each of the six *tiles* is encoded into a separate visual embedding vector for perception.

present in each tile, proving particularly useful in scenarios where the robot encounters feature-poor, uniformly colored objects such as walls, doors, or furniture. In such instances, an **abnormally** low standard deviation suggests that the vision encoder's output may lack reliability.

4.2. Navigability Middleware: Vision-Language Correlation

To distinguish navigable spaces from non-navigable ones, we designed a set of *positive prompts* describing clean and navigable environments, such as: "`A photo of a` **`(flat|open|wide|clear) {floor|ground|hallway}`**", and a set of *negative prompts* describing spaces that are cluttered by obstacles, such as: "`A` **`[cropped|bad|imcomplete]`** `photo of a` **`(blocked|messy|cluttered) {scene|space}`**" and "`A photo of a` **`(large|way blocking) {object|item}`**". As shown in Fig. 6 a, a clear floor is identified as navigable space, while the cluttered scene in Fig. 6c is accurately categorized as non-navigable.

For target discovery, a similar set of text prompts is used to define the *target* of a task. In our experiments, we use a toy bear (see Fig. 10 b) as the discovery target due to its uniqueness in the scene. We design a set of prompts that describe the target, *e.g.* "`A photo of a` **`(brown|toy) {bear|teddy bear}`**". We also design a set of nega-

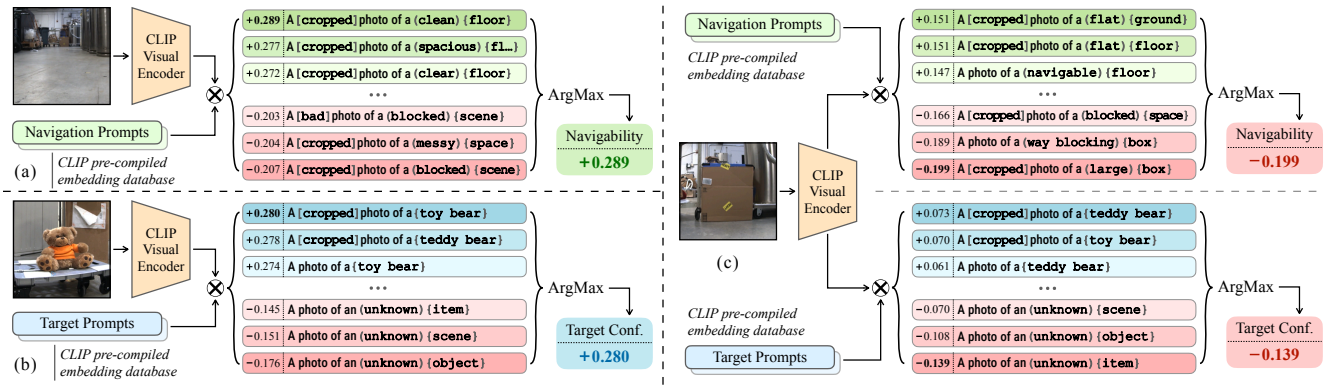


Figure 6. Detailed examples of the proposed *correlation middlewares* are illustrated; the circled cross symbol denotes the inner product of broadcasted vectors. (**a**) A navigable clean floor is encoded and correlated with the *navigability* database, where green rows (positive prompts) yield higher scores than red rows (negative prompts); the resulting positive final score indicates the space is navigable. (**b**) A toy bear (the target) is encoded and correlated with the target database, where blue rows (positive prompts) produce higher scores than red rows (negative prompts); the resulting positive final score confirms the target's presence. (**c**) A paper box is encoded and correlated with both databases; it corresponds to neither a navigable space nor the target, both scores are negative, indicating the space is not navigable and no target is present. [Best viewed digitally at 2× zoom.]

tive prompts that describe generic objects to filter out false positives, *e.g.* "`A photo of an` **`(unknown) {item|scene|object}`**". As shown in Fig. 6 b, the *target prompt* accurately identified the toy bear, while negative prompts effectively suppressed false positives on unrelated objects, such as the paper box in Fig. 6 c.

The symbols used in examples above belong to a custom designed prompt system. The same set of notations are used throughout this paper. The symbols are explained in detail in Sec.2 of the supplementary materials.

The resulting scores for both navigability and target confidence were computed on a per-tile basis. As depicted in Fig. 6 a-c, the CLIP visual encoder generated embeddings for each tile, which were then compared against a prompt database using inner products. The prompt database consisted of pre-encoded text prompts generated by the CLIP text encoder, organized into positive and negative categories. The resulting scores, ranging from -1.0 to 1.0, were determined by the highest absolute score among the prompt matches.

By employing both positive and negative prompts in each database and selecting the final result based on their contrast, the correlation process becomes more robust against fluctuations in absolute correlation values. This approach is particularly beneficial in environments with varying lighting conditions or complex scenes, where all scores may drift upward or downward depending on the quality of the visual input. Moreover, this method eliminates the need for manually setting a fixed threshold, further enhancing its adaptability.

4.3. Familiarity Middleware: Vision-Vision Correlation

In addition to navigability scores, a *familiarity database* is accumulated in real-time to track previously explored spaces. It is constructed with visual embeddings (512 dimensional vectors) representing known spaces without storing or using actual images. An incoming visual embedding is considered "known" if its correlation score with an existing vector exceeds a predefined threshold. Each new embedding vector is incrementally merged into the familiarity database; we implement the following two strategies for this:

1. Averaging among all vectors that belong to a known point, this involves keeping track of the count of vectors already merged into a known spot (s):

$$v_{\text{next}} = \frac{s}{s+1} \cdot v_{\text{prev}} + \frac{1}{s+1} \cdot v_{\text{new}}$$

2. Performing a rolling average operation upon merging a new vector; this method does not need to keep track of the total count of already merged vectors. Therefore, the vector tends to lean towards newly inserted vectors and gradually "forget" older ones. The "rate of forgetting" can be controlled by a factor λ (a.k.a decay factor):

$$v_{\text{next}} = (1-\lambda) \cdot v_{\text{prev}} + \lambda \cdot v_{\text{new}}$$

When no *known vector* exists in the database, the incoming vector is inserted as a new data point. Eventually, a familiarity score is generated for each perception vector, guiding navigation by encouraging the robot to prioritize unexplored areas over revisiting familiar ones.

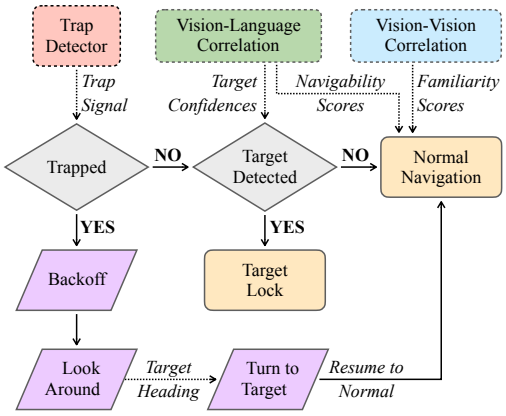


Figure 7. A state diagram illustrating the decision-making process, providing a detailed view of Fig. 5c; blocks with dashed borders are other computational blocks in Fig. 5. Dashed arrows represent data/signal propagation, while solid arrows denote conditions and transitions between states.

4.4. Navigation Decision Backend

Lastly, the decision module generates motion commands according to the information provided by the perception and correlation systems. Specifically, a "motion mixer" is introduced as the baseline correlation-to-motion translator. It takes all aforementioned scores for each tile (*i.e.*, *navigability*, *familiarity*, and *standard deviation*) into consideration and makes an intelligent decision. The motion mixer generally prioritizes highly navigable yet less familiar locations while avoiding areas with minimal texture, indicated by low standard deviation values. To handle non-trivial scenarios, the decision module incorporates two additional functionalities: (1) *trap detection*, enabling the robot to identify and escape from potential dead-ends, and (2) *look-around*, allowing it to reorient itself in complex environments. The overall decision-making process and state transitions are illustrated in Fig. 7.

Trap Detection. In certain scenarios, the robot may encounter a "dead end", where no navigable path is visible within its FOV. In rare instances, the vision-language model may generate false positive `Nav scores` for scenes it does not adequately comprehend. For example, the lack of salient features can lead to incorrect positive scores when the camera is positioned too close to a plain white wall. Such situations are defined as "trapped" states, which can occur under two conditions: (a) the proximity switch asserts a halt signal for a specified duration, or (b) the cumulative travel distance, as measured by odometry, falls below a defined threshold over a given period. Empirically, the system flags the robot as trapped if it travels less than 0.2 meters within the past 5 seconds.

Look Around. Due to the limited FOV from a single camera, the robot could only see objects in front of it, limiting the amount of information available to the robot for efficient navigation and exploration. To address this, a "look around" mechanism (see Fig. 8) is introduced to enable the robot to gain situational awareness by performing a $360°$ rotation while collecting `Nav scores` associated with different headings. A Gaussian convolution is then applied to these scores to identify the most navigable direction. When recovering from a "trapped" state, the *look around* mechanism prioritizes a direction different from the original heading by a linear factor k, rewarding headings that deviate from the initial orientation. Additionally, a *look around* behavior is triggered at the start of a new mission to ensure the robot identifies the most promising path for exploration.

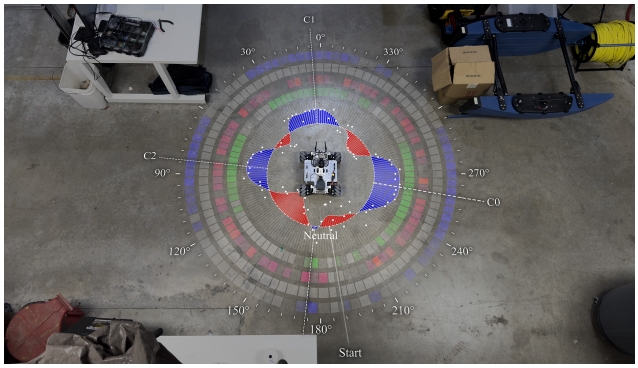


Figure 8. Rendering of a *look around* operation overlaid on a birds-eye view of the robot; it was able to identify navigable paths (blue bars) apart from obstacles (red bars) based on the proposed visual perception pipeline. The candidate headings are annotated as C_i. After the look-around operation, the robot selected C_0 as its next heading according to the area of free space. Details on the color codes and symbols are in Fig. 12.

5. Experimental Results and Analyses

5.1. Experimental Setup

Real-world experiments were conducted in two different environments as depicted in Fig. 10 a. The environments were selected for their cluttered layout and complex details, which include numerous obstacles, potential traps, and loops, providing a challenging setting for comprehensive analysis. As shown in Fig. 10, the robot was tasked with exploring the space while searching for a designated target – a toy bear approximately 20 cm tall and 10 cm wide, chosen for its distinctive appearance within the test scene. For each task, the source (robot's starting position) and destination (target location) were selected from predefined locations as labeled in Fig. 10 (b) and (d).

5.2. Performance Evaluation

Evaluation criteria. Our experimental trials are designed to evaluate *efficiency* of exploring an area and *success rates* of finding a target with no prior map or knowledge about the

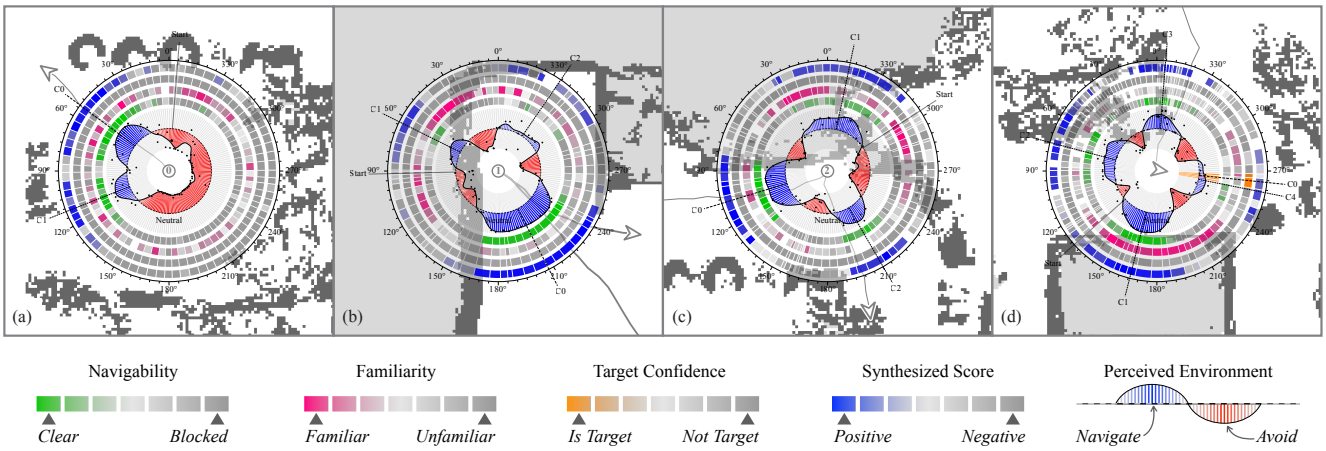


Figure 9. Visualizations of look-around operations during a demonstrative task; candidate directions are depicted as dashed lines (C_i), where candidates with smaller indices are assigned a higher priority. The plots are generated from correlation scores without utilizing additional sensory data. (**a**) Initial look-around at the center of the map: the system does not incorporate familiarity scores since the familiarity database is uninitialized; thus, the navigability score predominantly influences the robot's decision. (**b**) The robot encountered a dead-end and was temporarily trapped; a look-around operation enabled it to identify a navigable path and resume exploration. (**c**) The robot was trapped due to a false positive navigability perception caused by a transparent object; with a look-around, the system successfully recovered from the false positive and continued its exploration. (**d**) The target was located nearby in this scenario; through the look-around, the robot was able to identify the target and assign it as the highest-priority candidate. [Best viewed digitally at 2× zoom for clarity.]

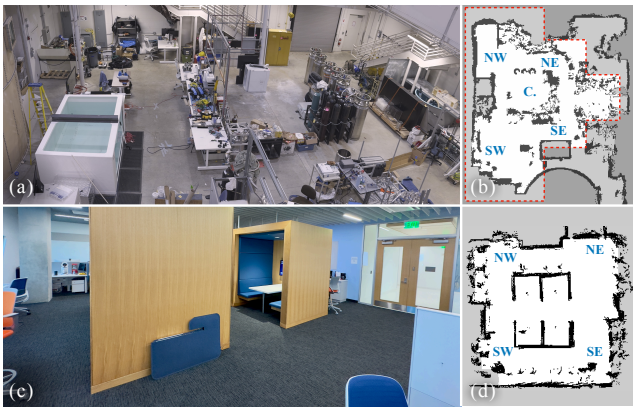


Figure 10. The environment used for real world exploration and discovery experiments: (**a**) Overview of the machine hall taken from the east side. (**b**) 2D map of the experimental space, with task locations annotated in blue; the area utilized for the experiments is enclosed within the dashed red line. (**c**) An office scene taken from the S. E.corner; (**d**) 2D map of the office scene, with task locations annotated in blue.

target. To quantify these, we use the total distance traveled (trajectory length) before approaching the target as the core metric. Instead of the total duration of the task, the travel distance metric is agnostic to the CPU/GPU performance of the onboard computer and the capability of the mechanical driving system, which are considered external factors that can be improved independently.

Algorithms for comparison. We compare the performance of the proposed VL-Explore system with six widely used map-traversal and path-finding algorithms. For fair comparisons, a novel 2D simulation framework, **RoboSim2D**, is developed. It utilizes 2D LiDAR maps from recorded scans of the same environment used in real-world experiments. Additionally, the size of the simulated robot was configured to match the dimensions of the physical robot, ensuring consistency across simulations and the real platform. See the supplementary materials (Sec. 5) for more details on the algorithmic implementation and simulation results.

Criteria of failure. Due to practical limitations, a mission is considered a failure if the distance traveled exceeds a predefined upper limit. For real-world experiments conducted in the test area (approximately 12×16 m), this limit is set to 100 meters. In simulations, the limit is set to $1,000$ meters for random walk, wall-bouncing, and wave-front algorithms. For Bug algorithms, loop detection is employed to identify endless loops, which are classified as failures. Since failed missions yield infinite travel distances, they are excluded from the metrics presented in Table 2, they are instead reflected in the overall success rate of an algorithm.

Evaluation metrics. As observed in our simulation results, for a given amount of information, the success rate R is inversely proportional to the normalized path length L. This relationship is expressed by a commonly adopted performance metric for VLN systems, known as *Success weighted by Path Length* (SPL) [3], defined as:

$$\text{SPL} = \frac{1}{N}\sum_{i=1}^{N} S_i \frac{l_i}{\max(p_i, l_i)}. \quad (1)$$

Note that, for each source-target combination, the first contact distance of WaveFront simulation is used as the baseline distance D_{baseline}. With the choice of $l = D_{\text{baseline}}$, we have $p_i > l, \forall p_i \in \{p\}$; hence, the above equation is simplified to:

$$\text{SPL} = \frac{1}{N} \sum_{i}^{S_i=1} \frac{D_{\text{baseline}}}{p_i} = \frac{N_s}{N} \sum_{i}^{S_i=1} \frac{L_i}{N_s} = R \cdot \overline{L} \quad (2)$$

$$\text{s.t.} \begin{cases} L_i = \dfrac{D_{\text{baseline}}}{p_i} & - \text{ Inverse Relative Distance} \\ N_s = \sum_i^{S_i=1} 1 & - \text{ Number of Success Runs} \\ R \ = \dfrac{N_s}{N} & - \text{ Success Rate} \\ \overline{L} = \sum_i^{S_i=1} \dfrac{L_i}{N_s} & - \text{ Mean Inverse Path Length} \end{cases} \quad (3)$$

As a consequence, the SPL metric equates to an inverse proportional relation between two quantities, namely the success rate R and the mean inverse path length $\overline{L}$, as:

$$\text{SPL} = \overline{L} \cdot R. \quad (4)$$

However, experimental data reveal that the curvature of the SPL function does not align well with the observed R–$\overline{L}$ curve, shown later in the experimental results (Fig. 11). That is, for the same exploration algorithm, setting a different failure criterion will result in a different SPL score. To better model the efficiency of an algorithm from observed data, **we propose a new metric "*Entropy Preserving Score*" (EPS)** to represent the amount of information available to a given algorithm. It has a range of $[0, 1]$. When $\text{EPS} = 0$, the algorithm operates without environmental information and lacks sensing capabilities beyond collision detection, as seen in methods like random walk and wall bounce. As EPS approaches 1, it gains more information or has higher environment sensing capability, and the performance is expected to increase accordingly. Considering the impact of EPS, the revised equation expands to:

$$f_1(\text{EPS}) = f_2(\overline{L}) \cdot f_3(R) \quad (5)$$

$$\text{s.t. } f_n(x) = k_n \cdot x^{\,p_n} + t_n$$

where $H_n = [k_n \ \ t_n \ \ p_n]$ are hyper-parameters specific to an environment. *i.e.* A specific environment can be characterized using the collection of 9 hyperparameters:

$$H = \begin{bmatrix} H_1 \\ H_2 \\ H_3 \end{bmatrix} = \begin{bmatrix} k_1 & t_1 & p_1 \\ k_2 & t_2 & p_2 \\ k_3 & t_3 & p_3 \end{bmatrix}. \quad (6)$$

These hyper-parameters are fitted to the $\overline{L} - R$ curve derived from random-walk simulation results in the given map, plus an additional boundary condition $f_1(1.0) = f_2(1.0) \cdot f_3(1.0)$, which is the ideal case when the algorithm has all information available and achieves a perfect success rate with optimal travel distance.

5.3. Qualitative and Quantitative Analyses

We conduct extensive real-world experiments with over 60 trials based on various combinations of source-target locations shown in Fig. 10. For the proposed VL-Explore system, three trials were performed for each origin-target pair. In comparison, 200 trials were recorded for each random walk experiment, while 180 evenly distributed headings were recorded for each combination of the wall bounce runs.

Samples of the trajectories traversed by VL-Explore and other algorithms are presented in Fig. 12, overlaid on a 2D map of the environment. Under *normal navigation mode*, VL-Explore exhibits a human-like motion strategy by navigating near the center of open spaces, enhancing exploration efficiency and reducing the likelihood of becoming trapped by obstacles. In contrast, path-finding algorithms often follow the contours of obstacles due to their lack of semantic scene understanding. During *look-around operations* (see Fig. 8 and Fig. 9), VL-Explore accurately discriminates between navigable spaces and obstacles. Additionally, it demonstrates a preference for unexplored (unfamiliar) areas over previously visited (familiar) regions, contributing to improved efficiency compared to traditional map-traversal algorithms.

Out of 60 trials in the industrial scene, 3 failure cases were recorded, yielding a 95% overall success rate for VL-Explore. For the office scene, only 1 failure case was recorded out of 12 trials, yielding a 91.7% success rate. Among the failure cases, two were caused by the trajectory length exceeding the limit, while the other two were caused by the robot getting jammed by an obstacle, which was not detected by either the VLN pipeline or the LiDAR proximity switch.

The quantitative results categorized by source and target locations are presented in Table.1 of the supplementary materials. To ensure comparability across different source-target pairs, the travel distance for each row is normalized by the baseline travel distance and then aggregated into Table 2. For randomized algorithms, the success rate R is a configurable hyperparameter. Sample results for $R = 50\%$ and $R = 80\%$ are provided to represent their underlying performance in Table 2, with the corresponding equipotential curves previously shown in Fig. 11. In contrast, deterministic algorithms (such as Bug variants) have a fixed success rate. Their performances are reported directly in Table 2 and visualized in Fig. 11 for comparison.

As these results demonstrate, with the same amount of information, the proposed system outperforms traditional map traversal algorithms by significant margins in success

Table 2. Quantitative performance of VL-Explore and other algorithms in comparison. Metrics used are defined in Eq. 3. The acronyms: S.R. - Success Rate R; P.L. - Mean Inverse Path Length $\overline{L}$.

Method	Large Size Industrial Scene				Medium Size Office Scene				Additional Requirements
	S.R.	P.L.	SPL	$\mathcal{E}$	S.R.	P.L.	SPL	$\mathcal{E}$	
★ **VL-Explore**	95.0%	0.50	**0.48**	**0.40**	91.7%	0.67	**0.62**	**0.51**	Monocular RGB Camera
Random Walk	50.0%	0.15	0.08	0.00	50.0%	0.18	0.09	0.00	Basic Collision (obstacle) Detection
	80.0%	0.11	0.09	0.00	80.0%	0.13	0.11	0.00	
Wall Bounce	50.0%	0.17	0.09	0.01	50.0%	0.23	0.11	0.02	
	80.0%	0.12	0.10	0.01	80.0%	0.16	0.13	0.02	
Wave Front	50.0%	0.36	0.18	0.11	50.0%	0.44	0.22	0.15	
	80.0%	0.27	0.22	0.11	80.0%	0.34	0.27	0.15	
Bug 0	45.0%	0.89	0.49	0.45	37.5%	0.81	0.39	**0.37**	Target Location
Bug 1	100.0%	0.29	0.29	0.19	100.0%	0.23	0.23	0.08	Target Location, Memory of trajectory
Bug 2	80.0%	0.68	**0.55**	**0.52**	100.0%	0.41	**0.41**	0.25	Target Location, `m-line` detection
Previous Work (scores reported from original paper)									
VLMnav	50.4%	0.42	0.21	N/A					● Online Reasoning, Depth Camera, Voxel Map
	33.2%	0.41	0.14	N/A					○ Online Reasoning (without nav components)
GOAT	83.0%	0.77	**0.64**	N/A					● Depth Camera, Long Term Memory, Onboard SLAM
	61.0%	0.31	0.19	N/A					○ Depth Camera, Onboard SLAM (first-time exploration)

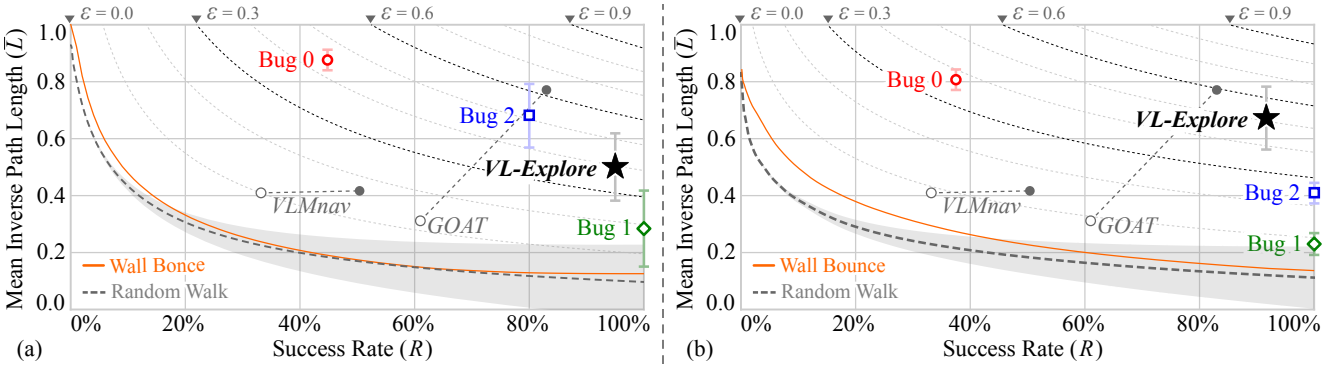
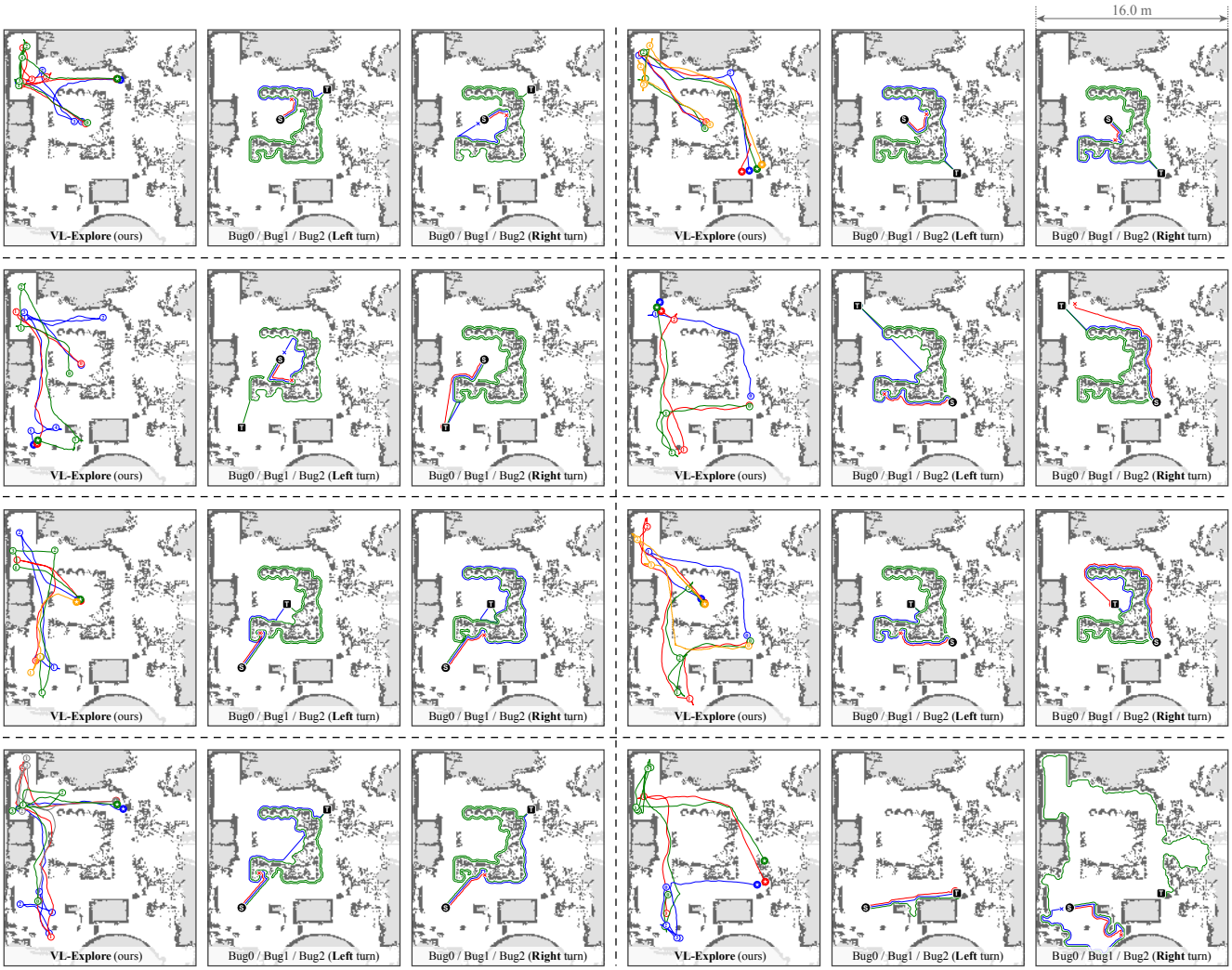


Figure 11. Performance of VL-Explore compared to map-traversal methods (*i.e.* random walk, wall bounce) [34], path-finding (Bug) algorithms [28, 29] and SOTA VLN methods (*i.e.* VLMnav and GOAT) [6, 13] for UGV navigation. SOTA VLN methods are represented by a solid gray dot (all systems enabled) and a gray circle (certain systems disabled to match with information required by VL-Explore). Detailed explanations are provided in Table.2. (a) Large industrial scene; (b) Mid-size office scene. The equipotential lines labeled as EPS $= x$ are defined by Eq. 5.

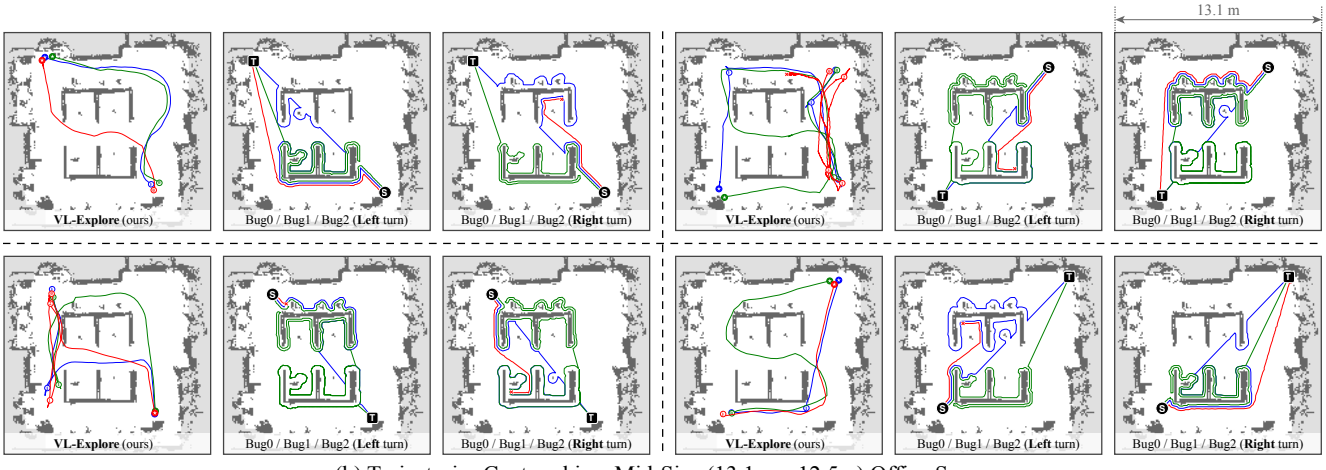
rates, trajectory lengths, SPL and EPS scores. Despite the inherent disadvantage of operating without prior target knowledge or precise localization, the system achieves performance comparable to path-finding algorithms. Notably, in many scenarios, it surpasses path-finding algorithms in either trajectory efficiency or success rates. These results highlight the effectiveness of the proposed vision-language based pipeline for simultaneous exploration and target discovery tasks.

5.4. Comparative Analyses of SOTA VLNs

We also compare VL-Explore with SOTA VLN systems based on their features and prerequisites. It is important to note that, due to different experimental setup and reporting criteria, a perfect quantitative comparison is not always feasible. For example, SEEK [12] reported 0.65 ± 0.35 SPL for random walk, while our random walk simulation yielded 0.08 ± 0.11 SPL, showing a magnitude of difference. Preliminary analysis suggests that this discrepancy stems from SEEK's reliance on a node-graph map structure, which reduces the problem space to a finite set of graph nodes and edges. In contrast, VL-Explore operates continuously in an unknown environment, resulting in a substantially larger problem space, facilitating active robot navigation with no prior information or pre-compiled waypoints. This fundamental difference motivated us to introduce the EPS metric, which helps reduce the discrepancies caused by varying prerequisites and experimental setups.



(a) Trajectories Captured in a Large (16m × 18m) Insdutrial Scene



(b) Trajectories Captured in a Mid-Size (13.1m × 12.5m) Office Scene

Figure 12. Sample results of autonomous exploration and target discovery by VL-Explore compared to Bug algorithms. The trajectories of each algorithm are overlaid on a 2D map of the test environment (north up). Circled numbers indicate the *look around* operations of VL-Explore and the corresponding waypoints sequence it followed. For Bug algorithms, the red, green, and blue trajectories represent paths traversed by *Bug0*, *Bug1*, and *Bug2*, respectively. Best viewed digitally at 2× zoom for clarity; more results and analyses are included in Sec.6 of the supplementary materials.

Among the vast volume of prior attempts in the field of vision-language based navigation or exploration, we select two recent representative systems that provide ablation results with similar prerequisites as our proposed system:

- VLMnav [13] is an end-to-end VLN system, which assumes no prior knowledge or map; it does not rely on online SLAM for navigation. However, it requires a depth camera to obtain navigability scores (floor mask). It reported 0.210 SPL on the ObjectNav dataset, its score drops to 0.136 without the depth camera.
- GOAT [6] is a lifelong mapping system; it requires an RGBD camera to obtain necessary information for navigation and map construction. In addition to the final target, it also takes "sequential goals" that help it to take intermediate moves. It reported 0.64 SPL on the "in the wild" dataset with persistent memory, but the score drops to 0.19 in the absence of prior knowledge. The authors did not perform an ablation study when depth sensing is removed.

In contrast, our proposed work, *VL-Explore*, achieves **0.357** SPL without requiring depth sensing, persistent memory, or prior knowledge (*e.g.* mapping run) of a scene. To the best of our knowledge, VL-Explore is the **first VLN system** to achieve efficient indoor visual exploration **without any depth or range information** (*e.g.* RGBD, multi-camera, LiDAR). VL-Explore holds a significant advantage in its intended use case – first-time exploration and target discovery with a single monocular camera.

There also exist many other works based on different prerequisites, which cannot be directly included in quantitative comparison. We instead summarize the characteristics of each system in Table 3. Compared to existing systems, VL-Explore uniquely requires zero prior knowledge of the task space and operates solely with monocular RGB perception. Additionally, contemporary VLN systems do not directly produce low-level motion commands but instead generate high-level instructions that depend on another traditional localization and navigation pipeline for the eventual motion execution. In contrast, VL-Explore directly outputs low-level motion commands, eliminating the need for auxiliary control and localization systems. This feature makes VL-Explore particularly suitable for rapid deployment on low-cost robots with limited computational and sensing resources.

5.5. Ablation Study

In order to understand the contribution of each middleware on the overall performance, we conduct an ablation study by disabling one middleware at a time and re-evaluating the system's performance. The experiment is performed in the medium-sized office scene shown in Fig. 10 (c), with the robot starting from the South-West (SW) corner and tasked to find the target in the North-East (NE) corner.

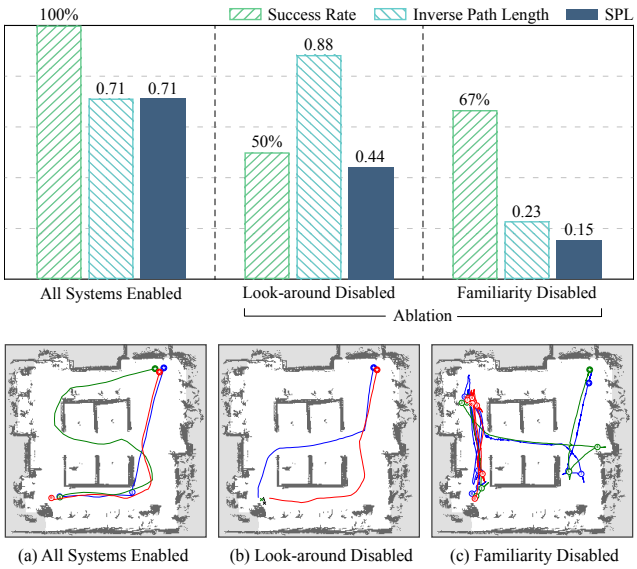


Figure 13. Ablation study of the proposed *VL-Explore* system; (a) Comparative results with all systems enabled; (b) Results with the *look-around* subsystem disabled, the robot either completes the task with no turn-around, or directly fail when being trapped, yielding a higher PL but a lower SR; (c) Results with the *familiarity* subsystem disabled, the robot tends to revisit previously explored areas, resulting in longer trajectories and lower success rates.

- **Disabling the *look-around* action.** In this part, the system does not perform any look-around action during a task, including the initial look-around that helps the robot finding its optimal initial heading. The initial orientation of the robot is manually set to a different direction for each trial. The results are shown in Fig. 13b. With the lack of look-around action, the task would fail upon encountering a dead-end as it could not recover from such a situation. This resulted in a significantly lower success rate (50%). However, in the cases when the robot successfully navigated to the target, the reported inverse path lengths turned out better due to survivorship bias. *i.e.* inefficient runs tend to fail instead of being recovered by a look-around action.
- **Disabling the *familiarity* middleware.** In this part, the *familiarity* middleware is disabled by truncating the familiarity scores to a fixed value. The results are shown in Fig. 13c. Without the familiarity middleware, the robot exhibited a strong tendency to revisit previously explored areas. In this particular case, the robot tends to run back and forth between S.W. and N.W. corner of the scene. This lead to significantly longer trajectory lengths (*i.e.* smaller mean inverse path length), and also induced negative impact on the overall success rate.

To summarize, the additional system components added to the VL-Explore pipeline are proven to be essential for its

Table 3. A qualitative comparison of features and prerequisites among SOTA VLN systems, highlighting the unique advantage of the proposed *VL-Explore* system that requires no extra sensor modalities or any separate path-finding or localization steps.

VLN System	Focus	Prerequisites	Sensing Modality	Output Type	Deployment	Evaluation
CLIP-Nav [9]	Scene	Pre-captured Images of candidate locations	N/A	Choice of an Image for Next Step	Simulation	Success rate (relative)
CLIP on Wheels [10]	Object	Unspecified	RGB-D Camera	Relevance map of the current View	Simulation	Accuracy & Path length
SEEK [12]	Object	Pre-built dynamic scene graph and relational semantic network	LiDAR, 3 RGB Cameras Real-time SLAM	Sequence of Waypoints	Real World	Success rate & Path length
GCN for Navigation [23]	Object	Pre-constructed knowledge graph of object relations	Monocular RGB Camera	Choice of a Pre-encoded Action	Simulation	Path length & Success rate
Chen *et al*. [8]	Scene	Pre-constructed node graph of navigable space	N/A	Choice of Next Node	Simulation	Success rate & Completion Rate
VL-Explore ★	Hybrid	None	Monocular RGB Camera	Direct Motion Commands (3_{DOF})	Real World	Path length & Success rate

overall performance. In addition to the basic obstacle avoidance and target identification capabilities provided by the vision-language correlation middlewares, the *look-around* action helps the robot to optimize its decision and recover from trap conditions, and the *familiarity* middleware ensures efficient exploration by constantly steering the robot into unexplored area.

6. Limitations, Failure Cases, and Potential Improvements of VL-Explore

6.1. Texture-less or Transparent Surfaces

As a vision-driven system, the proposed VL-Explore pipeline encounters challenges in environments with texture-less or transparent surfaces. When presented with a blank or non-informative image, the inherent VLM struggles to generate meaningful responses, disrupting the entire pipeline. For instance, as demonstrated in Fig. 9 d (C1), the robot mistakenly classified a water tank with transparent walls as navigable space. This error occurred because the VLM perceived the interior of the tank, visible through its transparent walls, as accessible terrain. To mitigate this issue, a standard deviation threshold was introduced to filter out ambiguous tiles. While this approach improved performance, the system still exhibits a general tendency to misclassify transparent or uniformly colored surfaces, such as blank walls or white floors, as navigable space. These challenges persist, particularly when the robot is positioned close to such surfaces, where the lack of texture further confuses the VLM. Note that this issue with transparent and texture-less walls has been an open problem for 2D indoor navigation by mobile robots [44].

6.2. Open Space and Artificially Constructed Mazes

The system's reliance on the semantic understanding capabilities of the VLM limits its effectiveness in large, open spaces. Without sufficient visual cues or meaningful objects in the environment, the robot cannot optimize its exploration based on semantic information. In such cases, the robot defaults to moving straight ahead until it encounters visually significant objects, such as obstacles, at which point its correlation middleware resumes providing meaningful guidance. This limitation reduces the system's efficiency in environments devoid of distinguishing visual features.

Moreover, our experiments in artificially constructed mazes, such as those built from paper boxes, revealed limitations in the VLM's ability to differentiate between the maze walls, such as monochrome cardboards and floors. As a result, the robot either failed to navigate or attempted to collide with the maze walls. This behavior can be attributed to the lack of similar examples in the VLM's training data, which likely did not include artificially constructed environments of this nature. Consequently, the system is better suited to real-world scenarios where the VLM has been trained on relevant, diverse visual data. This observation suggests that our proposed system will work effectively in real-world environments where VLMs generally work well.

6.3. Familiarity Saturation

The *familiarity* middleware is designed to help the robot avoid revisiting already explored areas by accumulating memory of the environment. However, during extended missions, the familiarity database tends to saturate due to the repetitive nature of objects in the environment. Once all unique objects are recorded, the system struggles to prioritize unexplored areas effectively. Future iterations of the system could implement a memory decay mechanism, allowing older entries in the familiarity database to fade over time, thereby prioritizing new observations and maintaining exploration efficiency.

6.4. Adaptive Slicing Strategies

In our current implementation of VL-Explore, a fixed slicing strategy is used based on the camera frame's aspect ratio, which generally performs well in general. However, in specific scenarios involving narrow navigable spaces (*e.g.*, a 50 cm gap between two obstacles), the robot often fails to recognize these spaces as traversable. This is because the narrow passage does not occupy an entire tile, leaving obstacles visible in all tiles and leading the system to avoid the space rather than navigate through it. Future improvements could include the development of advanced slicing strategies that dynamically adapt the *region of interest* based on the surroundings. This could be achieved using sparse object detection or pixel-wise segmentation models, enabling better handling of narrow spaces.

6.5. High Level Reasoning & Task Narration

One of the most promising extensions of this work lies in incorporating high-level reasoning and task narration using an integrated LLM. With this addition, the robot could leverage common-sense reasoning to make more intelligent navigation decisions. For example, when prompted to locate an item in a different room, the robot could infer that heading toward a door is the most logical action. Moreover, in bandwidth-limited applications, the LLM could provide real-time task narration by interpreting visual embeddings to describe mission progress. This approach would significantly reduce the communication bandwidth required compared to streaming raw video. By addressing these limitations and pursuing the outlined improvements, the proposed system can evolve into a more robust, efficient, and versatile platform for VLN in complex real-world environments.

7. Conclusion

In this paper, we introduced *VL-Explore*, a novel navigation pipeline designed for simultaneous exploration and target discovery by autonomous ground robots in unknown environments, leveraging the power of VLMs. Unlike traditional approaches that decouple exploration from path planning and rely on inefficient algorithms, *VL-Explore* enables **zero-shot inference** and efficient navigation using only monocular vision, without prior maps or target-specific information. To validate our approach, we developed a functional prototype UGV platform named *Open Rover*, optimized for general-purpose VLN tasks in real-world environments. Extensive evaluations demonstrated that *VL-Explore* outperforms state-of-the-art map traversal algorithms in efficiency and achieves comparable performance to path-planning methods that rely on prior knowledge. The integration of a CLIP-based VLM into a real-time navigation system underscores the potential of *VL-Explore* to advance intelligent robotic exploration. As a modular and flexible framework, *VL-Explore* lays the groundwork for future research in applying VLMs to more complex and dynamic robotic applications such as autonomous warehousing, security patrolling, and smart home assistance.

Acknowledgments

This work is supported in part by the National Science Foundation (NSF) grant #2330416; the Office of Naval Research (ONR) grants #N000142312429, #N000142312363; and the University of Florida (UF) ROSF research grant #132763

References

[1] Nour AbuJabal, Mohammed Baziyad, Raouf Fareh, Brahim Brahmi, Tamer Rabie, and Maamar Bettayeb. A comprehensive study of recent path-planning techniques in dynamic environments for autonomous robots. *Sensors*, 24(24), 2024. ISSN 1424-8220. doi: 10.3390/s24248089. URL https://www.mdpi.com/1424-8220/24/24/8089.

[2] Robin Amsters and Peter Slaets. *Turtlebot 3 as a Robotics Education Platform*, pages 170–181. Springer International Publishing, 01 2020. ISBN 978-3-030-26944-9.

[3] Peter Anderson, Angel X. Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, and Amir R. Zamir. On evaluation of embodied navigation agents. *CoRR*, abs/1807.06757, 2018. URL http://arxiv.org/abs/1807.06757.

[4] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments, 2018. URL https://arxiv.org/abs/1711.07280.

[5] Rodrigue Bonnevie, Daniel Duberg, and Patric Jensfelt. Long-term exploration in unknown dynamic environments. In *2021 7th International Conference on Automation, Robotics and Applications (ICARA)*, pages 32–37. IEEE, 2021.

[6] Matthew Chang, Theophile Gervet, Mukul Khanna, Sriram Yenamandra, Dhruv Shah, So Yeon Min, Kavit Shah, and Chris Paxton et. al. GOAT: GO to Any Thing, 2023. ArXiv: 2311.06430.

[7] Boyuan Chen, Zhuo Xu, Sean Kirmani, Brain Ichter, Dorsa Sadigh, Leonidas Guibas, and Fei Xia. Spatialvlm: Endowing vision-language models with spatial reasoning capabilities. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14455–14465, 2024.

[8] Kevin Chen, Juan Pablo de Vicente, Gabriel Sepulveda, Fei Xia, Alvaro Soto, Marynel Vazquez, and Silvio Savarese. A behavioral approach to visual navigation with graph localization networks. In *Proceedings of Robotics: Science and Systems*, FreiburgimBreisgau, Germany, June 2019. doi: 10.15607/RSS.2019.XV.010.

[9] Vishnu Sashank Dorbala, Gunnar Sigurdsson, Robinson Piramuthu, Jesse Thomason, and Gaurav S Sukhatme. CLIP-nav: Using CLIP for zero-shot vision-and-language navigation. In *Workshop on Language and Robotics at CoRL 2022*, 2022.

[10] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 23171–23181, 2022. doi: 10.1109/CVPR52729.2023.02219.

[11] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. Clip on wheels: Zero-shot object navigation as object localization and exploration. *arXiv preprint arXiv:2203.10421*, 3(4):7, 2022.

[12] Muhammad Fadhil Ginting, Sung-Kyun Kim, David D. Fan, Matteo Palieri, Mykel J. Kochenderfer, and Ali akbar Aghamohammadi. SEEK: Semantic reasoning for object goal navigation in real world inspection tasks. In *Proc. of Robotics: Science and Systems*, 2024.

[13] Dylan Goetting, Himanshu Gaurav Singh, and Antonio Loquercio. End-to-end navigation with vlms: Transforming spatial reasoning into question-answering. In *Workshop on Language and Robot Learning: Language as an Interface*, 2024.

[14] Tianrui Guan, Yurou Yang, Harry Cheng, Muyuan Lin, Richard Kim, Rajasimman Madhivanan, Arnie Sen, and Dinesh Manocha. Loc-zson: Language-driven object-centric zero-shot object retrieval and navigation. *ArXiv*, abs/2405.05363, 2024. doi: 10.48550/arXiv.2405.05363.

[15] D. Hahnel, R. Triebel, W. Burgard, and S. Thrun. Map building with mobile robots in dynamic environments. In *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, volume 2, pages 1557–1563 vol.2, 2003. doi: 10.1109/ROBOT.2003.1241816.

[16] Carlos Alberto Velásquez Hernández and Flavio Augusto Prieto Ortiz. A real-time map merging strategy for robust collaborative reconstruction of unknown environments. *Expert Systems with Applications*, 145:113109, 2020.

[17] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10608–10615. IEEE, 2023.

[18] Gabriel Ilharco, Mitchell Wortsman, Ross Wightman, Cade Gordon, Nicholas Carlini, Rohan Taori, Achal Dave, Vaishaal Shankar, Hongseok Namkoong, John Miller, Hannaneh Hajishirzi, Ali Farhadi, and Ludwig Schmidt. Openclip, July 2021. If you use this software, please cite it as below.

[19] Jongheon Jeong, Yang Zou, Taewan Kim, Dongqing Zhang, Avinash Ravichandran, and Onkar Dabeer. Winclip: Zero-/few-shot anomaly classification and segmentation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19606–19616, June 2023.

[20] Russell Keith and Hung Manh La. Review of autonomous mobile robots for the warehouse environment. *arXiv preprint arXiv:2406.08333*, 2024.

[21] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. In *Proceedings. 1985 IEEE International Conference on Robotics and Automation*, volume 2, pages 500–505, 1985. doi: 10.1109/ROBOT.1985.1087247.

[22] Hyungseok Kim, Hyeongjin Kim, Seonil Lee, and Hyeonbeom Lee. Autonomous exploration in a cluttered environment for a mobile robot with 2d-map segmentation and object detection. *IEEE Robotics and Automation Letters*, 7(3): 6343–6350, 2022.

[23] D. A. Sasi Kiran, Kritika Anand, Chaitanya Kharyal, Gulshan Kumar, Nandiraju Gireesh, Snehasis Banerjee, Ruddra dev Roychoudhury, Mohan Sridharan, Brojeshwar Bhowmick, and Madhava Krishna. Spatial relation graph and graph convolutional network for object goal navigation, 2022. URL `https://arxiv.org/abs/2208.13031`.

[24] Jacob Krantz, Erik Wijmans, Arjun Majumdar, Dhruv Batra, and Stefan Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. *ArXiv*, 2020.

[25] Xiwen Liang, Liang Ma, Shanshan Guo, Jianhua Han, Hang Xu, Shikui Ma, and Xiaodan Liang. Cornav: Autonomous agent with self-corrected planning for zero-shot vision-and-language navigation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 12538–12559, 2024.

[26] Iker Lluvia, Elena Lazkano, and Ander Ansuategi. Active mapping and robot exploration: A survey. *Sensors*, 21(7): 2445, 2021.

[27] Yuxing Long, Wenzhe Cai, Hongcheng Wang, Guanqi Zhan, and Hao Dong. Instructnav: Zero-shot system for generic instruction navigation in unexplored environment. *arXiv preprint arXiv:2406.04882*, 2024.

[28] Vladimir Lumelsky and Alexander Stepanov. Dynamic path planning for a mobile automaton with limited information on the environment. *IEEE transactions on Automatic control*, 31(11):1058–1063, 1986.

[29] Vladimir J Lumelsky and Alexander A Stepanov. Path-planning strategies for a point mobile automaton moving amidst unknown obstacles of arbitrary shape. *Algorithmica*, 2(1):403–430, 1987.

[30] Ji Ma, Hongming Dai, Yao Mu, Pengying Wu, Hao Wang, Xiaowei Chi, Yang Fei, Shanghang Zhang, and Chang Liu. Doze: A dataset for open-vocabulary zero-shot object navigation in dynamic environments. *arXiv preprint arXiv:2402.19007*, 2024.

[31] Steve Macenski and Ivona Jambrecic. Slam toolbox: Slam for the dynamic world. *Journal of Open Source Software*, 6 (61):2783, 2021. doi: 10.21105/joss.02783.

[32] James Clerk Maxwell. A dynamical theory of the electromagnetic field. *Philosophical Transactions of the Royal Society of London*, 155:459–512, 1865.

[33] Anwesan Pal, Yiding Qiu, and Henrik Christensen. Learning hierarchical relationships for object-goal navigation. In *Conference on Robot Learning*, pages 517–528. PMLR, 2021.

[34] Bao Pang, Jiahui Qi, Chengjin Zhang, Yong Song, and Runtao Yang. Analysis of random walk models in swarm robots for area exploration *. In *IEEE TRANSACTIONS ON EMERGING TOPICS IN COMPUTATIONAL INTELLIGENCE*, pages 2484–2489, 12 2019. doi: 10.1109/ROBIO49542.2019.8961844.

[35] Yan Peng, Dong Qu, Yuxuan Zhong, Shaorong Xie, Jun Luo, and Jason Gu. The obstacle detection and obstacle avoidance algorithm based on 2-d lidar. In *2015 IEEE international conference on information and automation*, pages 1648–1653. IEEE, 2015.

[36] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[37] Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. In *Conference on robot learning*, pages 492–504. PMLR, 2023.

[38] Jian Sun, Jie Zhao, Xiaoyang Hu, Hongwei Gao, and Jiahui Yu. Autonomous navigation system of indoor mobile robots using 2d lidar. *Mathematics*, 11(6):1455, 2023.

[39] Allen Taflove and Susan C. Hagness. *Computational electrodynamics: the finite-difference time-domain method*, volume 67–106. Artech House, 2nd edition, 06 2000. ISBN 1-58053-076-1.

[40] Chaoqun Wang, Wenzheng Chi, Yuxiang Sun, and Max Q-H Meng. Autonomous robotic exploration by incremental road map construction. *IEEE Transactions on Automation Science and Engineering*, 16(4):1720–1731, 2019.

[41] Wei Yang, Xiaolong Wang, Ali Farhadi, Abhinav Gupta, and Roozbeh Mottaghi. Visual semantic navigation using scene priors. *arXiv preprint arXiv:1810.06543*, 2018.

[42] Naoki Yokoyama, Sehoon Ha, Dhruv Batra, Jiuguang Wang, and Bernadette Bucher. Vlfm: Vision-language frontier maps for zero-shot semantic navigation, 2023. URL `https://arxiv.org/abs/2312.03275`.

[43] Qianfan Zhao, Lu Zhang, Bin He, Hong Qiao, and Zhiyong Liu. Zero-shot object goal visual navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2025–2031. IEEE, 2023.

[44] Chen Zhou, Jiaolong Yang, Chunshui Zhao, and Gang Hua. Fast, accurate thin-structure obstacle detection for autonomous mobile robots. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 1–10, 2017.

[45] Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. Esc: Exploration with soft commonsense constraints for zero-shot object navigation. In *International Conference on Machine Learning*, pages 42829–42842. PMLR, 2023.

[46] Jiawen Zhu and Guansong Pang. Toward generalist anomaly detection via in-context residual learning with few-shot sample prompts. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2024.

Appendix

A. Frame Slicing Strategies

To enhance the embedding of positional information within the pipeline, we preprocess the raw camera frames by slicing them into smaller *tiles* before inputting them into the vision-language model. This slicing strategy is specifically designed to optimize the robot's ability to make informed navigation decisions based on its visual perceptions. As illustrated in Fig. 1 in the main paper, the slicer divides each frame into six (2 rows $\times$ 3 columns) tiles, with the number of horizontal divisions aligned with the robot's control capabilities. Using smaller tiles instead of the full camera frame reduces visual distractions, thereby improving the accuracy of the correlation results.

The slicing dimensions are consistently maintained across all stages of the pipeline. For instance, under the 2×3 slicing strategy, a frame divided into six tiles produces a corresponding $2 \times 3 \times 512$ matrix of visual embeddings. These embeddings are subsequently aggregated and passed to the correlation middleware, which generates a 2×3 matrix of correlation scores, with each element corresponding to a specific tile. The influence of individual tiles on one another is determined by the logic of the respective correlation middleware. For example, while the familiarity middleware disallows correlations between tiles within the same frame, it may permit correlations between the lower-left tile of a previous frame and the upper-right tile of the current frame.

To preserve spatial context in each tile, sliced regions are slightly expanded beyond their original boundaries. This overlap introduces approximately 20% shared area between adjacent tiles, mitigating the loss of contextual information. This adjustment is beneficial when the camera encounters texture-less surfaces, such as walls, paper boxes, or door panels – where the absence of distinctive features could otherwise impair the robot's perception and navigation performance.

B. Text Prompt Generation

We also design a prompt system to compose text prompts in a hierarchical template. The proposed design is inspired by WinCLIP [19, 46], an anomaly detection framework. Our prompt system consists of the following templates and notations, which we follow throughout the paper.

- **Top-level prompts** such as:
  - "`A` **`[desc.]`** `photo of a` **`(state)`** **`{object}`**"
  - "`A` **`[desc.]`** `image of a` **`(state)`** **`{object}`**"
  - "`A` **`(state)`** **`{object}`**"
- **`[descriptions]`** such as **`clear`**, **`blurry`**, **`cropped`**, **`empty`**, **`corrupted`**, *etc.*
- **`(states)`** such as **`clean`**, **`clear`**, **`wide`**, **`narrow`**, **`cluttered`**, **`messy`**, **`way blocking`**, *etc.*
- **`{objects}`** such as **`floor`**, **`wall`**, **`door`**, **`object`**, *etc.*

We utilize a YAML-based syntax to define the structure of the prompt databases. YAML, a widely adopted and human-readable markup language, has been extended in our implementation to incorporate additional syntactical features for enhanced flexibility and fine-grained control over the generation of text prompt combinations. Two key features introduced are *selective integration* and *in-place expansion*.

1. **Selective integration** enables a prompt to bypass certain levels of the template hierarchy, thereby allowing precise customization. For instance, a prompt such as "`A` **`(meaningless)`** `photo`" should not be followed by any **`{object}`**. This is achieved by terminating the prompt early, omitting the insertion marker (*i.e.* **`{}`**) in its definition.
2. **In-place expansion** facilitates the collective definition of similar short prompts by using a vertical bar syntax (*i.e.* **`|`**) to separate terms. This feature is particularly effective when used in conjunction with *selective integration*. For example, the prompt "`A photo with no` **`{context|texture|information}`**" is expanded into three distinct prompts. Although not fitting into the hierarchical template structure, these can be defined as top-level negative prompts, aiding in distinguishing meaningful content from irrelevant or meaningless information.

These enhancements to the YAML-based syntax provide an effective framework for defining and managing prompt databases with high precision and adaptability.

C. Performance Optimization

Navigation decisions are a critical component of real-time autonomous robotic systems, requiring rapid execution to ensure safety and operational efficiency. However, most vision-language models are designed as extensions of LLMs, where processing delays and data throughput are less critical. To evaluate the suitability of VLMs for robotic navigation, we identify two key performance metrics:

1. **Decision delay** measures the time elapsed between capturing a frame from the robot's camera and issuing a corresponding motion command to the motors. This metric reflects the system's ability to promptly react to environmental changes, such as avoiding obstacles and maintaining safe navigation.
2. **Throughput** quantifies the number of frames processed per second, directly influencing the smoothness of the robot's motion. Limited computational resources neces-

Table 4. Performance margins of the navigation pipeline of *VL-Explore* at different optimization levels. The acronyms: **[P]** Preprocess, **[I]** Inference, **[C]** Correlation, and **[D]** Decision.

Config.	Mean Data Processing Delay (ms)					Throughput (mean FPS)
	[P]	**[I]**	**[C]**	**[D]**	Total	
CPU Seq.	80.57	2221.67	3.13	0.37	2534.55	0.41 ± 0.07
CPU Para.	73.16	2201.84	3.29	0.40	2463.26	0.44 ± 0.02
GPU Seq.	65.93	178.77	2.94	0.45	327.57	3.22 ± 0.25
GPU Para.	25.32	171.08	3.43	0.48	252.10	$\mathbf{5.01} \pm 0.36$

sitate dropping unprocessed frames, retaining only the latest frame for decision-making. Higher throughput minimizes inter-frame discrepancies, reducing abrupt changes in motion and ensuring smoother transitions. Mathematically, throughput is the inverse of the decision delay: *i.e.* $f = \frac{1}{T_{\text{delay}}}$.

On the other hand, most robotic systems rely on embedded computers featuring RISC architecture processors and power-limited batteries, prioritizing power efficiency over computational capability. Addressing these constraints, we developed an optimized architecture to maximize resource utilization while improving decision delay and throughput.

Focusing on reducing decision delay and increasing data throughput, our proposed framework divides the navigation pipeline into four major nodes: pre-processing, inference, correlation, and decision-making. Computationally intensive tasks are offloaded to the GPU to exploit parallel processing capabilities. These optimizations were implemented on a power-limited embedded system, balancing efficiency and performance. With the proposed optimizations, we achieve: (**i**) decision delay: $252.10\,\text{ms}$, a 900% improvement; and (**ii**) throughput: 5.01 FPS (frames per second), representing a 400% improvement compared to the sequential CPU-based implementation. These results, detailed in Table 4, demonstrate the viability of our VLM-based navigation pipeline for real-time robotic applications under computational constraints.

D. Usage of 2D LiDAR

The proposed *VL-Explore* pipeline uses only monocular RGB data for VLN navigation decisions. Nevertheless, we installed a 2D 360° scanning LiDAR on the robot for experimental safety, map generation, and comparative analyses. The specific LiDAR functions are listed below.

1. **Emulation of a proximity kill-switch.** The LiDAR serves as a virtual *kill switch* during operation, detecting obstacles in the robot's intended path. It monitors the motion commands sent to the wheels to infer the robot's planned trajectory and checks for potential obstructions along that direction. A *halt signal* is asserted upon detection of a potential collision, thus strictly used for safety purposes without influencing the navigation algorithm.
2. **Visualization of map and robot trajectory.** Mapping is performed offline using recorded LiDAR and odometry data, with subsequent trajectory analysis to evaluate navigation efficiency and overall performance. We utilize the *SLAM Toolbox* [31] to generate maps and trajectories, implementing a two-pass process to enhance the quality of the results. In the first pass, the SLAM Toolbox operates in offline mapping mode, creating a high-resolution 2D map of the environment. In the second pass, the pre-constructed map is loaded and run in localization-only mode to generate accurate trajectories using LiDAR localization. Despite this two-pass process, the trajectories generated often exhibit zig-zag patterns caused by a mismatch between SLAM drift correction frequency and the robot's odometry update rate. These patterns are not observed in the actual motion of the robot; still, they affect our performance evaluation due to increased lengths of the projected paths. To mitigate this, we introduce a *stride* parameter to smooth the trajectory. This smoothing parameter reduces the projected trajectory length and better matches the robot's actual travel distance.
3. **Comparison with simulated algorithms.** A 2D map generated using LiDAR data is employed to simulate and evaluate traditional range sensor-based map traversal and path-planning algorithms for performance comparison. The LiDAR data collected during *VL-Explore* experiments was used to reconstruct the exploration map, which is reused for simulations. This approach ensured that all comparisons were conducted on the same map, providing a fair and consistent evaluation of the different algorithms.

E. Simulation Algorithms

For map traversal algorithms [34], we consider:

1. **Random Walk:** Starts with a given heading, then randomly selects a new heading when an obstacle is encountered.
2. **Wall Bounce:** Starts at a given heading, then bounces off obstacles based on the normal vector of the impact point.
3. **Wave Front:** Starts as a Gaussian probability distribution, then spreads outwards and bounces off obstacles. A detailed explanation is provided below.

Additionally, we compare the following Bug Algorithms [28, 29]:

1. **Bug0:** Moves straight toward the target until encountering an obstacle, then follows the obstacle boundary until it can resume a direct path to the target.
2. **Bug1:** Heads towards the target when possible, otherwise, circumnavigates the obstacle. After looping an obstacle, travels to the point with the minimum distance on the loop.

3. **Bug2:** Circumnavigates obstacles upon encountering until it crosses the direct line from the start to the target (the `m-line`), then resumes a straight path toward the target.

E.1. Wave Front Simulation

The "Wave Front" simulation is a custom-developed package to serve as a general baseline metric for comparison between different methods. It utilizes the concept of classic wave function [32] and interprets it as a probability distribution, allowing simultaneous exploration of infinitely many directions driven by the following equation:

$$\frac{\partial^2}{\partial t^2}\,\psi(x,y,t) = c^2\nabla^2\psi(x,y,t) \tag{7}$$

The simulation implements a 2D Laplacian equation at discretized time steps. Given wave speed c, time step Δt, and spatial step Δx, the wave equation is discretized as:

$$\begin{aligned}\psi(x,y,t_{n+1}) = \; & 2\cdot\psi(x,y,t_n) - \psi(x,y,t_{n-1}) \\ + \; & \frac{c^2\Delta t^2}{\Delta x^2}\nabla^2\psi(x,y,t) \\ s.t. \quad & t_{n+1} - t_n = \Delta t\end{aligned}$$

Here, the discretized Laplacian operator ∇^2 [39] is defined as:

$$\begin{aligned}\nabla^2\psi(x,y) = \; & \psi(x_{i+1},\, y_j) + \psi(x_{i-1},\, y_j) \\ & + \psi(x_i,\, y_{j+1}) + \psi(x_i,\, y_{j-1}) \\ & - 4\cdot\psi(x,y) \\ s.t. \quad & x_{i+1} - x_i = y_{j+1} - y_j = \Delta x\end{aligned}$$

Besides, map boundaries and obstacles are simulated by reflective conditions. That is, at boundary point (x, y), we have: $\nabla\psi(x,y)\cdot\hat{\mathbf{n}}_{x,y} = 0$, where $\hat{\mathbf{n}}_{x,y}$ is the normal vector. In a discretized 2D grid, the directions of this normal vector are simplified to four possibilities, *i.e.*, $\hat{\mathbf{n}} \in \{\pm\hat{\mathbf{x}}, \pm\hat{\mathbf{y}}\}$.

To implement the aforementioned boundary condition, the discretized Laplacian operator can be rewritten as:

$$\begin{aligned}\nabla^2\psi(x,\,y) = \; & k_{i+1,\,j}\cdot\psi_{i+1,\,j} + k_{i-1,\,j}\cdot\psi_{i-1,\,j} \\ & + k_{i,\,j+1}\cdot\psi_{i,\,j+1} + k_{i,\,j-1}\cdot\psi_{i,\,j-1} \\ & - (k_{i+1,\,j} + k_{i-1,\,j} + k_{i,\,j+1} + k_{i,\,j-1})\cdot\psi(x,\,y)\end{aligned}$$

$$s.t. \quad k_{i,j} = \begin{cases} 1 & \text{if } (x_i,\, y_j) \text{ is free space} \\ 0 & \text{if } (x_i,\, y_j) \text{ is obstacle} \end{cases}$$

At time $T = 0$, a probability distribution is initialized as a Gaussian distribution centered at the robot's starting location, with a standard deviation set to half of the robot's size; the total probability is normalized to 1. At each iteration, the wave function is multiplied by an inverse Gaussian function centered around the target location, effectively reducing the total probability on the map and simulating a draining effect. The simulation terminates when the total remaining probability falls below a defined threshold ($1e^{-4}$ in our experiments). The outcome is a 2D probability distribution $p(t)$ indicating the likelihood of the robot reaching the target at a given time point. The mean and standard deviation of this distribution are used for comparison with other trajectory-based algorithms.

At a high level, this algorithm simulates the behavior of infinitely many wall-bouncing robots moving in all possible directions. Notably, the drain operation creates a uniform gradient toward the target, subtly guiding the robot. As a result, the algorithm performs slightly better than a purely randomized wall-bouncing simulation, which lacks any target information. The travel distance when the first non-zero probability is detected is used as the baseline distance for each origin-target combination. For relative performance comparisons, the travel distances are normalized by this baseline distance before aggregating the results.

F. Additional Experimental Results

The full set of trajectories recorded in the experiment is shown in Fig. 14 and Fig. 15. Metrics measured for each source-target combination are shown in Table 5.

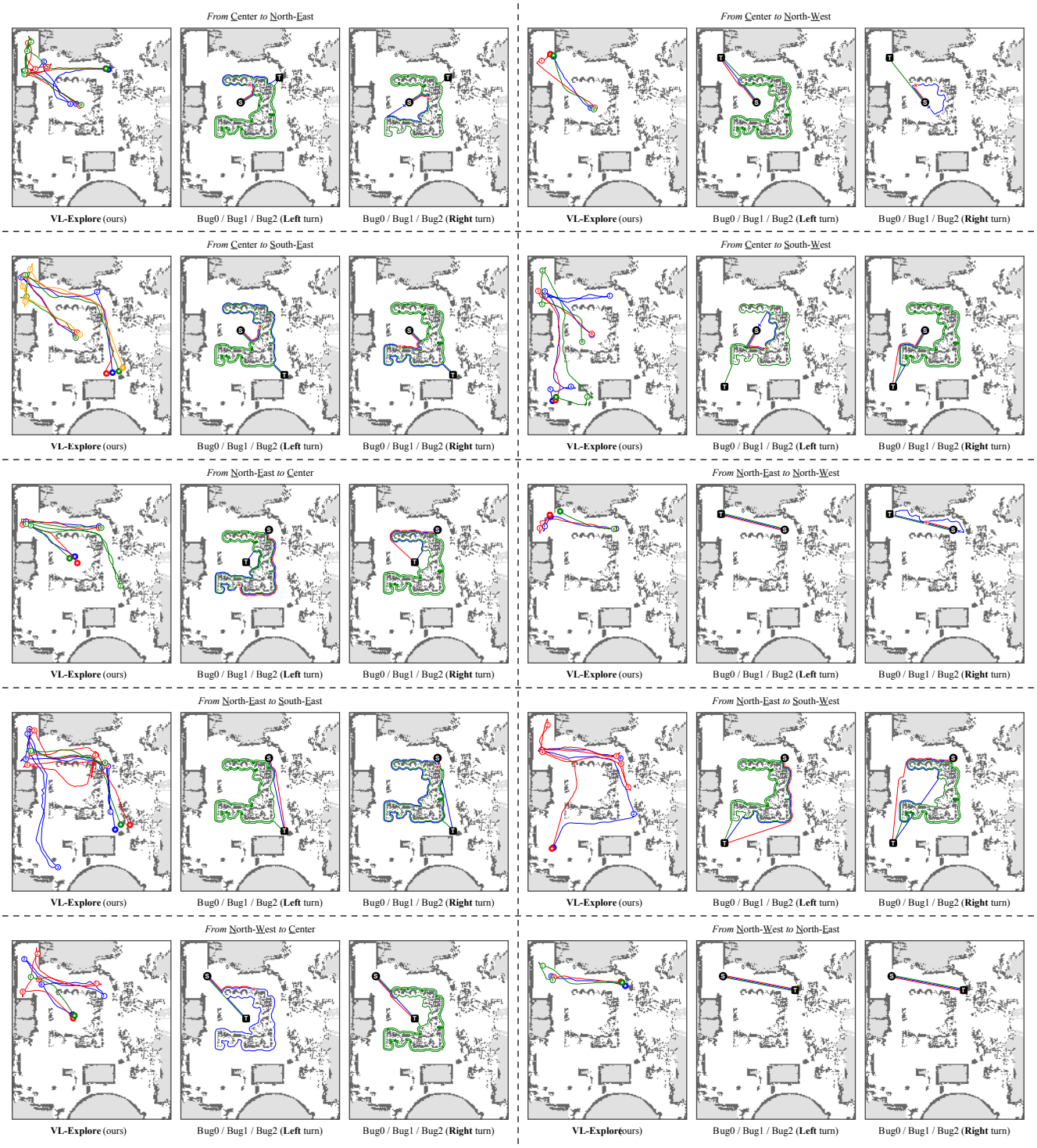


Figure 14. Trajectories of all results reported in the experiment section (part.1)

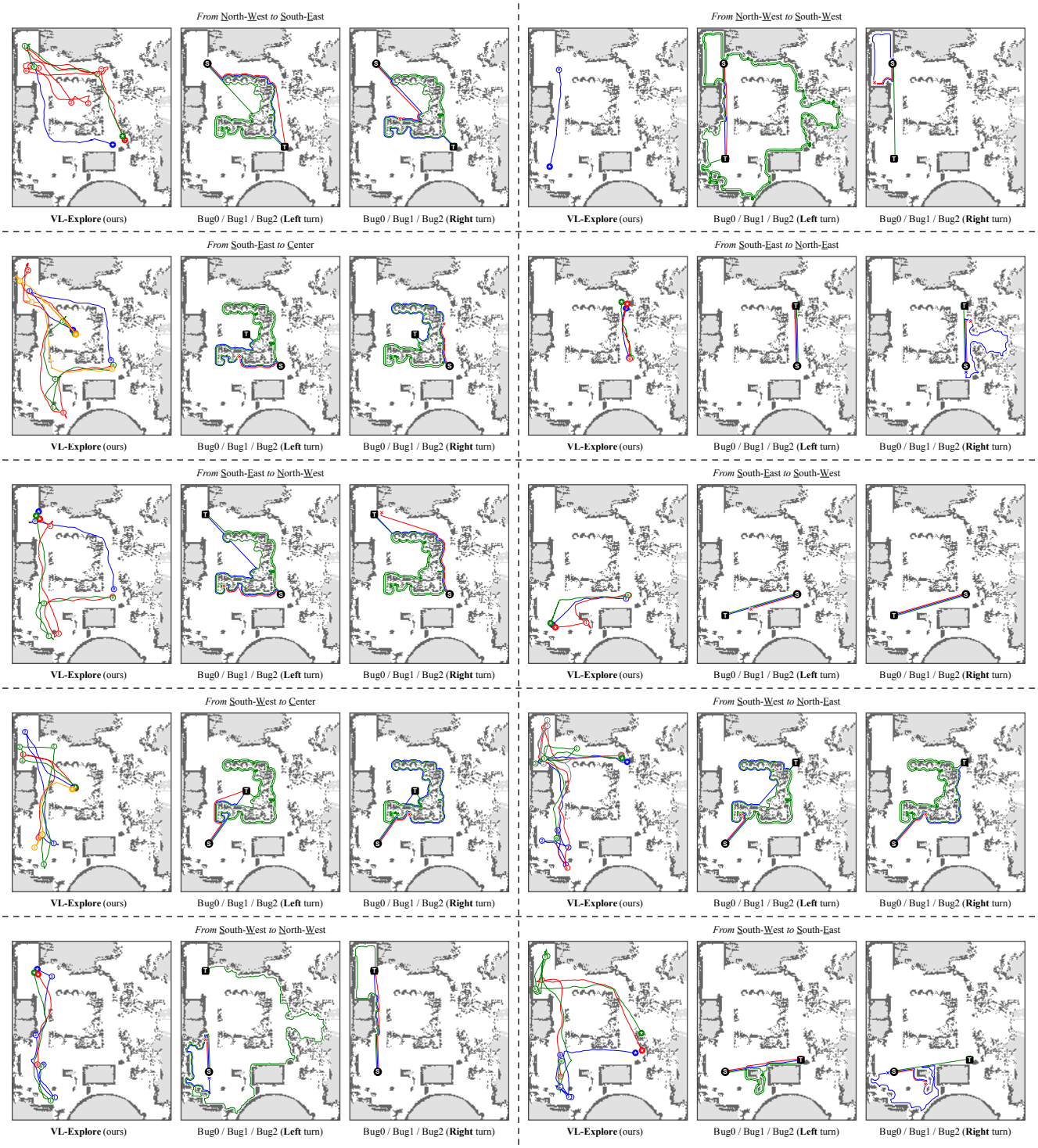


Figure 15. Trajectories of all results reported in the experiment section (part.2)

Table 5. Quantitative performance comparison of the exploration (random walk, wall bounce, and wavefront) and path-finding (Bug 0/1/2) algorithms in terms of trajectory length. All units are in meters; see Fig. 10 for the source-target layouts. The acronyms: **[L]** and **[R]** represent left and right turn rules, respectively; the failure cases are shown as '×' markers.

Task Setup		***VL-Explore*** ★	Random Walk	Wall Bounce	Wave Front	Bug0		Bug1		Bug2	
Source	Target	Avg. ± Std.	Avg. ± Std.	Avg. ± Std.	Avg. ± Std.	**[L]**	**[R]**	**[L]**	**[R]**	**[L]**	**[R]**
Warehouse Scene ($16m \times 18m$)											
C.	N.W.	8.3± 2.1	149.6±18.6	169.3±14.9	162.7±55.8	5.9	×	75.8	42.2	6.0	×
	N. E.	27.2± 2.7	326.5±27.0	313.1±19.4	247.4±74.3	×	×	63.0	49.5	13.4	×
	S.W.	29.0± 7.1	256.7±19.4	233.4±15.6	236.9±66.8	×	7.9	43.7	72.5	×	8.6
	S. E.	27.0± 3.7	370.6±23.4	338.4±18.8	285.2±82.5	×	×	54.9	59.2	23.2	19.4
N.W.	C.	21.6±10.9	233.3±26.0	193.6±16.2	192.2±55.8	×	5.9	47.0	67.7	40.1	6.0
	N. E.	8.7± 2.2	253.7±25.4	220.1±18.3	233.5±61.0	7.4	7.4	7.4	7.4	7.4	7.4
	S.W.	9.9± 0.0	290.2±21.7	241.5±14.6	239.4±69.0	9.6	×	132.1	84.4	10.0	×
	S. E.	29.6±17.9	338.8±24.9	332.7±19.9	281.5±78.7	14.8	×	63.8	52.0	15.7	24.8
N. E.	C.	23.6±12.1	275.6±24.5	251.2±20.0	223.4±62.6	×	9.2	50.5	60.9	28.0	12.0
	N.W.	9.1± 2.4	157.5±21.5	187.0±17.0	181.9±45.6	6.9	×	6.9	75.7	6.9	×
	S.W.	42.0± 9.5	236.2±19.3	250.9±14.2	237.9±70.5	14.3	13.9	58.9	55.2	17.1	24.0
	S. E.	48.7±19.4	235.8±22.5	277.9±18.2	240.1±67.9	7.6	×	68.3	44.7	7.7	41.0
S.W.	C.	19.3± 6.0	340.9±26.8	331.7±19.1	266.0±66.2	×	×	66.1	50.0	12.0	33.7
	N.W.	15.9± 4.5	329.3±22.6	330.6±17.2	259.9±63.8	×	10.5	65.1	19.7	×	10.9
	N. E.	31.1± 3.1	396.7±29.0	288.3±16.7	276.8±78.6	×	×	51.8	65.1	27.5	15.8
	S. E.	31.8±11.2	217.2±22.0	208.3±16.5	220.7±62.3	7.8	×	10.1	71.4	8.0	×
S. E.	C.	29.7± 8.0	386.0±25.9	311.1±17.7	283.3±72.9	×	16.1	57.2	54.5	18.2	21.2
	N.W.	22.5± 4.4	369.9±26.5	298.7±17.7	262.8±63.5	×	×	50.6	64.9	25.5	15.1
	N. E.	5.7± 0.2	302.3±28.8	279.7±20.1	249.3±60.8	6.1	×	6.1	73.4	6.1	×
	S.W.	11.1± 1.8	156.3±20.4	147.2±13.2	171.3±58.7	7.5	7.5	7.5	7.5	7.5	7.5
Office Scene ($13m \times 12.5m$)											
N.W.	S. E.	22.8± 3.6	173.6±15.7	148.0±10.6	48.0±22.2	×	×	88.4	64.1	25.4	34.9
N. E.	S.W.	36.3± 5.7	243.3±19.8	207.7±13.5	56.9±28.2	×	17.0	66.6	88.0	23.9	35.0
S.W.	N. E.	16.9± 4.3	263.9±24.4	268.3±16.9	65.7±34.3	×	14.5	39.2	47.0	36.0	31.4
S. E.	N.W.	14.5± 0.7	266.3±19.2	290.4±18.5	72.5±38.6	15.9	×	47.4	40.1	37.7	25.9