ATLAS: Adaptive Landmark Acquisition using LLM-Guided Navigation

Utteja Kallakuri
Johns Hopkins University
ukallak1@jh.edu

Bharat Prakash
University of Maryland, Baltimore County
bhp1@umbc.edu

Arnab Neelim Mazumder
University of Maryland, Baltimore County
arnabm1@umbc.edu

Hasib-Al Rashid
University of Maryland, Baltimore County
hrashid1@umbc.edu

Nicholas R. Waytowich
Army Research Laboratory
nicholas.r.waytowich.civ@army.mil

Tinoosh Mohsenin
Johns Hopkins University
tinoosh@jhu.edu

Abstract

Autonomous navigation agents traditionally rely on predefined maps and landmarks, limiting their ability to adapt to dynamic and unfamiliar environments. This work presents ATLAS, a novel system that continuously expands its navigable landmark set and performs complex natural language-guided navigation tasks. ATLAS integrates three key components: a path planning module for navigating to known landmarks, an object detection module for identifying and localizing objects in the environment, and a large language model (LLM) for high-level reasoning and natural language understanding. We evaluate ATLAS in diverse virtual environments simulated in Gazebo, including indoor office spaces and warehouses. Results demonstrate the system's ability to steadily expand its landmark set over time and successfully execute navigation tasks of varying complexity, from simple point-to-point navigation to intricate multi-landmark tasks with natural language descriptions. Our comprehensive tests show that ATLAS expands its knowledge, achieving a 100% success rate in tasks involving known landmarks and up to 100% in semantically inferred goals for objects not in the initial Knowledge Base. We further demonstrate the system's capacity to enhance its navigational knowledge incrementally, showcasing its ability to dynamically adapt and accurately perform complex, natural language-driven tasks in diverse simulation environments.

1. Introduction

Navigation tasks involving natural language instructions pose a significant challenge for autonomous agents [2, 4]. While traditional navigation systems rely on predefined maps and landmarks, real-world environments are dynamic and often require the ability to continuously learn and adapt. This paper presents a novel approach that combines various modules, including natural language processing, object detection, and path planning, to create an intelligent navigation system capable of continuously expanding its knowledge base and performing complex navigation tasks.

The motivation behind this work stems from the observation that humans excel at navigating unfamiliar environments by continuously updating their mental maps and exploiting their ability to understand natural language descriptions. For instance, when exploring a new city, humans can identify notable landmarks through visual cues and incorporate them into their mental map [7, 19]. Furthermore, they can follow directions like "Head towards the tall clock tower, then turn left at the fountain" by grounding the language instructions to the perceived landmarks. This remarkable ability to fuse perception, language understanding, and spatial reasoning enables humans to navigate efficiently in novel environments [10].

The proposed ATLAS system aims to emulate this human-like navigation capability by integrating key components that enable continuous learning and reasoning. It is designed to address three objectives: 1) continually growing the set of navigable landmarks, 2) performing navigation tasks involving one or more landmarks, and 3) utilizing a large language model (LLM) to assist with high-level reasoning, semantic navigation, and landmark set expansion [6, 22]. By leveraging these capabilities, the system bridges the gap between natural language instructions and physical navigation in dynamic environments. In this work we use Cohere as the LLM [5].

At the core of ATLAS is a navigation module that can

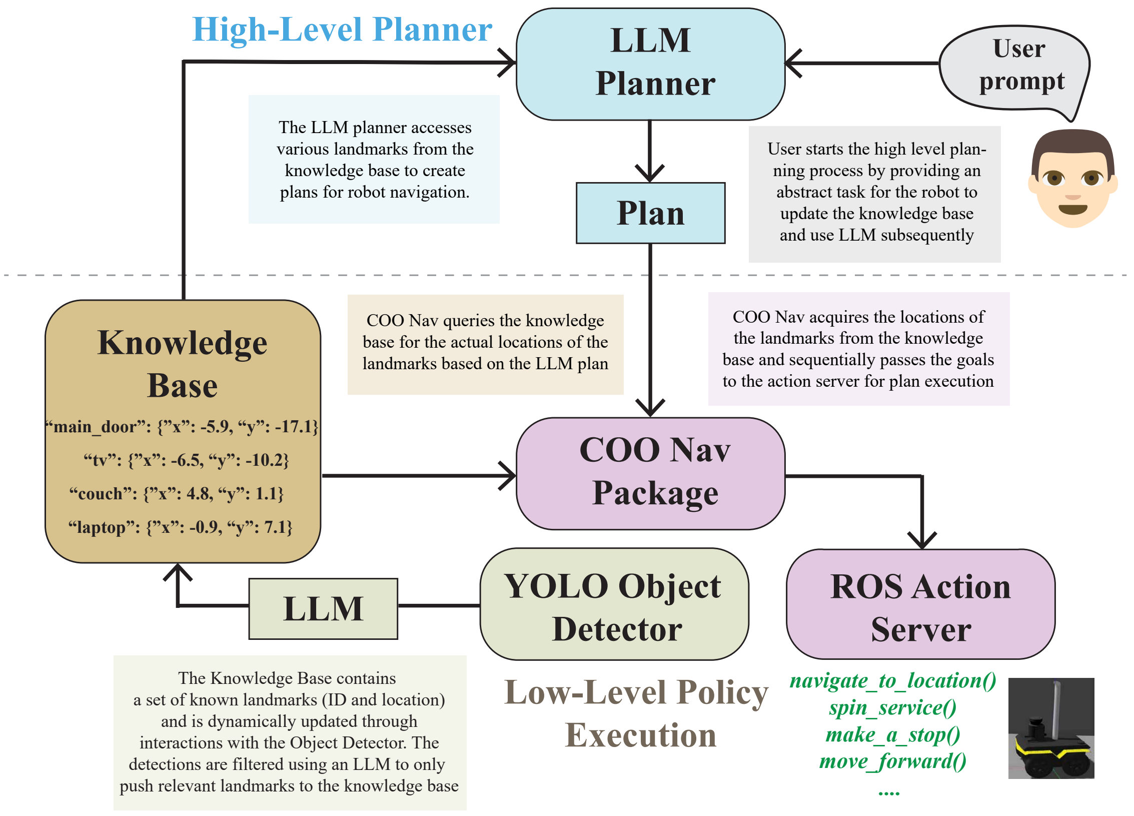
High-Level Planner
LLM Planner
User prompt
The LLM planner accesses various landmarks from the knowledge base to create plans for robot navigation.
Plan
User starts the high level planning process by providing an abstract task for the robot to update the knowledge base and use LLM subsequently
Knowledge Base
"main_door": {"x": -5.9, "y": -17.1}
"tv": {"x": -6.5, "y": -10.2}
"couch": {"x": 4.8, "y": 1.1}
"laptop": {"x": -0.9, "y": 7.1}
COO Nav queries the knowledge base for the actual locations of the landmarks based on the LLM plan
COO Nav acquires the locations of the landmarks from the knowledge base and sequentially passes the goals to the action server for plan execution
COO Nav Package
LLM
YOLO Object Detector
ROS Action Server
The Knowledge Base contains a set of known landmarks (ID and location) and is dynamically updated through interactions with the Object Detector. The detections are filtered using an LLM to only push relevant landmarks to the knowledge base
Low-Level Policy Execution
navigate_to_location()
spin_service()
make_a_stop()
move_forward()
....

Figure 1. High-level system architecture detailing the components and data flow between the LLM Planner, COO Navigation Package, ROS Action Server, Knowledge Base with Landmarks, and Object Detector. The agent is setup with an initial set of landmarks it can navigate to in its knowledge base. The agent receives a goal which is parsed by the LLM and looks up the knowledge to get navigation goals. This is then used by the ROS planner for navigation. The agent detects objects along the way which is then filtered through the LLM to get a relevant objects. These new objects are stored back in the knowledge base. The dictionary progressively keeps growing as the agent explores the environment.

plan paths to known landmarks based on their coordinates. However, the true power of ATLAS lies in its ability to learn and expand its knowledge through an object detection module and an LLM [24]. The detection module enables the agent to identify objects in its environment, while the LLM provides high-level reasoning and natural language understanding capabilities.

The integration of these components allows ATLAS to continuously grow its landmark set by interpreting natural language instructions, detecting and localizing new objects, and incorporating them as navigable landmarks. Furthermore, the LLM's semantic understanding capabilities enable ATLAS to perform complex navigation tasks involving multiple landmarks and high-level reasoning, such as interpreting and executing instructions like "Go to the kitchen and then proceed to the living room."

The key contributions are summarized as follows:

- A system architecture that integrates a path planning module, object detection, and a large language model for navigation tasks.
- A novel method for continuously expanding the set of navigable landmarks by detecting objects in the environment and incorporating them into the system's knowledge base with the assistance of the LLM.
- An approach to performing complex navigation tasks involving multiple landmarks and high-level reasoning by leveraging the LLM's semantic understanding capabilities.
- Experimental evaluation demonstrating the system's performance in various navigation scenarios, highlighting its adaptability, continuous learning, and execution of complex tasks with natural language instructions.

2. Background and Related Works

Edge deployment of large multimodal models [15–18] along with the intersection of natural language understanding and robotics have led to the field of Vision-and-Language Navigation (VLN) [2, 14], where the integration of linguistic instructions with visual perceptions aids autonomous agents navigate and interact within an environment. The advent of Large Language Models (LLMs) has introduced new dimensions to decision-making in VLN, diverging into two main trajectories: utilizing LLMs as auxiliary helpers or primary planners.

In the domain of using LLM as an auxiliary helper, works such as LM-Nav [20] leverage LLMs to generate contextual information, such as landmark lists from raw navigation instructions, aiding vision-language models in inferring joint probability distributions. These applications underscore LLMs' potential beyond mere information generation, suggesting their capability for comprehensive planning grounded in real-world environments.

LLM as a primary planner domain explores LLMs' capability to craft executable plans, directly or through low-level planners, for embodied agents [1, 8, 9, 13, 21, 25]. While these investigations underline the feasibility of LLM-driven planning, they often rely on the premise of predefined admissible actions, a condition that could limit practical applicability due to the dynamic nature of real-world environments and the increasing complexity with the number of environment objects.

Voyager [23] is one such work in which the authors present an LLM-powered agent in Minecraft, autonomously exploring and learning complex skills through a novel curriculum, skill library, and a code refinement process, showcasing unmatched proficiency and adaptability. ITP [12] presents a framework that uses language models for robots to perform tasks and adapt to new goals. ITP leverages LLMs for high-level planning and execution, integrating planning, vision, and skill execution. GOAT [3] is a universal navigation system for mobile robots with three key contributions: a multimodal approach for handling goals via category labels, images, and language descriptions; a lifelong learning capability that utilizes past environmental experiences for improved navigation; and a platform-agnostic design enabling deployment across various robotic platforms.

Our paper introduces a novel integration of natural language processing, object detection, and a LLMs. Unlike ITP[12], which relies on predefined guidelines for task execution, our approach dynamically expands navigable landmarks through environmental interaction and language model insights, facilitating continuous learning. Contrary to VOYAGER's[23] focus on skill acquisition within simulated settings, our method emphasizes real-world landmark learning for navigation. Furthermore, our system employs a language model for semantic reasoning and complex task execution, a capability not present in GOAT[3]. This integration enables sophisticated navigation based on natural language instructions, showcasing our system's adaptability and superior performance in real-world scenarios.

3. Proposed Approach

We begin the proposed approach by describing the problem statement and the overall architecture shown in Figure 1.

3.1. Problem Statement

We consider a system where the agent receives instructions in the form of natural language describing a navigation task. The agent has a navigation module which is capable of planning a path to a landmark provided it has the corresponding co-ordinates on the map. The agent is initialized with a small set of landmarks to which it can navigate. It is also equipped with an object detection module that can detect objects as well as their distances from its current position. Finally, the agent also has access to an LLM to help with high-level reasoning.

Given these ingredients, our objective is to build a system which can 1) continually grow the set of landmarks it can navigate to, 2) perform navigation tasks that involve one or more landmarks. 3) Use the LLM to assist with navigation, grow the landmark set, and perform semantic navigation.

3.2. System Architecture

The agent architecture consists of three main modules. First, a path planning module called the COO Nav (Coordinate Navigation), which is built on top of ROS. Given a map, this module is used to perform path planning and navigate to a landmarks given a set of co-ordinates. Second, the high level planner module is an LLM which parses a natural language input and provides a structured plan. The plan comprises of a sequence of waypoints or landmarks that the agent must navigate to. The prompt to the LLM is constructed such that it has access to the current set knowledge base(KB) of landmarks and the output is a subset of landmarks required to complete the given task.

Third, the object detection module is responsible to detecting and localizing objects in the environment which is then used to acquire more landmarks in the environment. This module uses a YOLO-v5s model [11]—to identify new objects as the agent is navigating the environment. This coupled with a depth sensor is used to localize new objects in the environment. However, the object detector is likely to detect random objects that might not make sense as landmarks. Instead of using a human to filter theses, we use another instance of the LLM to perform this task. Since LLMs capture human intuition, we design a prompt that asks the LLM to decide if it makes sense to add an object as a landmark. These new filtered landmarks are added to the KB which keeps on growing with time.

Given a natural language instruction, these three components act together to perform high level planning, navigate in the environment, and add new landmarks to the growing KB. This process is outlined in the algorithm 1.

3.3. Algorithmic Overview

The COO Nav package acting as the low-level policy bridges the gap between high-level plans generated by the LLM and the low-level ROS Action Server that controls UGV movement. COO Nav interfaces directly with ROS's actionlib to execute navigational tasks providing the services $navigate\_to\_location$ and $spin\_service$. The $navigate\_to\_location$ service controls the movement of the UGV to specified waypoints. It retrieves the coordinates from the KB based on the input location identifier and sends a goal to the ROS Action Server, directing the UGV to navigate to these coordinates. The $spin\_service$ on the other hand, ensures the robot is able to make a in-place rotation thereby scanning the environment for any objects of interest to be added into the KB.

The YOLO object detector is used for real-time object detection and executes callback functions to manage image and depth data from appropriate ROS topics and to update ATLAS with the latest inputs and detections. This operates in parallel with the navigation components and it continuously scans the UGV's surroundings to identify and classify objects which are used to enrich the KB. This ensures that ATLAS's environmental understanding evolves with each deployment, enhancing its navigational intelligence.

Algorithm 1 Dynamic Navigation and Knowledge Enrichment

```
Require: Small Predefined set of known locations in KB
 1: function NavigateToLocation($location$)
 2:     $coo = getCoordFromKB(location)$
 3:     $ros\_navigate(coo)$
 4:     $spin()$
 5:     return $success$
 6: end function
 7: function UpdateKB($detectedObjects$)
 8:     $relevantObjs = LLM\_filter(detectedObjects)$
 9:     for all $object$ in $relevantObjs$ do
10:         $objPos, objOrient = getPosOrient(object)$
11:         if object not in KB then
12:             Add $object, objPos, objOrient$ to KB
13:         end if
14:     end for
15: end function
16: Initialize COO Nav and YOLO Object Detector
17: while UGV is operational do
18:     Capture user prompt and parse via LLM Planner
19:     Translate LLM output to locations
20:     NavigateToLocation(location)
21:     $objs = detectObjects()$
22:     UpdateKnowledgeBase(objs)
23: end while
```

4. Experimental Setup and Results

4.1. Environment

To evaluate the performance of ATLAS, we conducted extensive experiments in diverse virtual environments using the Gazebo simulator. Gazebo provides a realistic 3D simulation of robotic environments, enabling us to rigorously test ATLAS's capabilities in controlled yet complex scenarios. We constructed multiple virtual worlds with varying landmark configurations, object placements, and environmental conditions, mimicking real-world settings such as indoor office spaces, residential spaces, and natural landscapes.

Within these simulated environments, we deployed a mobile robot equipped with sensors for perception and navigation, replicating the setup of ATLAS. The robot's perception module utilized a simulated RGB-D camera and a LiDAR sensor to detect and localize objects, while the navigation module planned and executed paths based on the available landmark information. The LLM operated in parallel, processing natural language instructions and assisting the robot in expanding its landmark knowledge and reasoning about complex navigation tasks. This experimental setup allowed us to systematically evaluate ATLAS's performance across a range of navigation scenarios, from simple point-to-point navigation to complex multi-landmark tasks with natural language descriptions.

4.2. Results

The first set of results illustrates ATLAS's capability to continuously grow its navigable landmark set over time. We present a graph that plots the number of landmarks in the robot's KB as a function of time spent exploring the virtual environment. The graph demonstrates a steady increase in the set of landmarks, showcasing ATLAS's ability to successfully detect, localize, and incorporate new objects as navigable landmarks with the assistance of the LLM. For the second set of results, we evaluate ATLAS's performance on various types of navigation tasks, ranging from simple point-to-point navigation to complex multi-landmark tasks with natural language descriptions. We report the task success rate, a metric that measures the percentage of navigation tasks successfully completed by the robot. We also test ATLAS's ability in semantic goal reaching which evaluates the agent's ability to leverage the language model's semantic understanding to interpret and execute navigation instructions involving landmarks not present in the current landmark set.

4.2.1 Growing Knowledge Base

The effectiveness of a fully autonomous system can be determined by its adaptive learning capabilities, specifically its ability to continuously integrate new environmental data into its navigational framework. In our case, this is captured in the expansion of the KB. The KB, initialized with a basic set of landmarks (referred to as 'KL_INIT'), is progressively augmented as the UGV explores its environment and identifies new objects of interest, as shown in Figure 2.

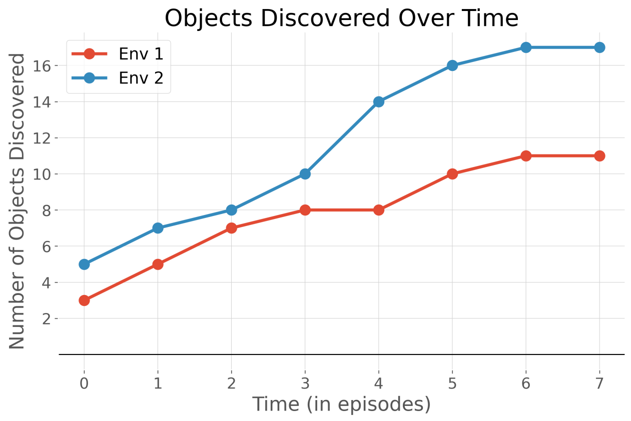

Figure 2. The total number of unique objects discovered by the UGV over time in two distinct environments. This shows the ability of the UGV to dynamically update its Knowledge Base with new landmarks as it navigates in the environments. Env 1 represents a residential setting, while Env 2 represents a warehouse setting.

The incremental discovery process is shown by the UGV's exploration in two different environments. In 'Env 1', a household environment, the UGV's KB grew from the initial three landmarks to a total of eleven. Each additional entry in the dictionary represents the inclusion of a newly identified landmark during a task execution, such as 'Go to Room 1' and 'Go to Room 2'.

For 'Env 2', a warehouse setting, a more complex environment, this KB growth is more evident. Here we start with five pre-known landmarks and extending to a comprehensive suite of seventeen. This environment has a broader array of object classes, showing the UGV's capacity to adapt its KB in diverse settings.

The growth of the KB over time has significant implications for the execution of navigational tasks. With an enriched set of landmarks, the UGV is more equipped to understand and navigate to new locations, facilitating a higher task success rate while enabling the high level planner (HLP) to make more efficient plans. The relation between the breadth of the KB and the UGV's navigational performance highlights ATLAS's ability to leverage its growing database for enhanced autonomous operation.

This dynamic learning ability, combined with the language model's semantic understanding, allows the UGV to extend its navigational capabilities beyond the initial landmark set, demonstrating ATLAS's potential in adaptive long-term deployment and unseen scenarios.

4.2.2 Task-Oriented Navigation and Incremental Learning in Varied Settings

Figure 3 presents a detailed visualization of an autonomous task execution within a residential setting by the UGV, resulting in the dynamic update of the KB. The task begins with a user prompt, which is sequentially processed by the high-level and low-level planning stages.

The process initiates with a user prompt:"Investigate Rooms 1 and 2". The HLP interprets this high-level command to provide a sequential plan to investigate two specific rooms within the residential environment. The HLP translates the user prompt into actionable goals for the UGV: first, to Go to Room 1, and then Go to Room 2. These goals are then broken down into discrete actions by the Low-Level Planner, which commands the UGV to move to the corresponding locations using the $navigate\_to\_location$ service and execute a $spin\_service$. This spin ensures that the UGV performs a 30° in-place rotation in the clock-wise and counterclockwise directions to scan the environment for any objects of interest.

As the UGV executes the planned navigation the perception component, identifies and localizes objects such as a refrigerator, suitcase, potted plant, couch, and laptop. These objects are highlighted by bounding boxes in the Camera Front View. Each detected object is assessed against the current KB; if an object is not already present in the KB and is within a specified distance threshold, it is added as a new landmark. This process is shown by the growing entries in the KB, where new objects like 'Refrigerator' and 'Laptop' are incorporated as the UGV explores Rooms 1 and 2. The Gazebo and camera views collectively illustrate the UGV's field of operation and its perception field, respectively. The integration of these views with the planning stages and the KB updates provides a comprehensive feedback loop that not only demonstrates the ATLAS's current operational state but also informs its future behavior by increasing its spatial understanding and navigational database.

Similarly, Figure 4 depicts task-oriented navigation in a warehouse setting. The sequence again starts with a user-defined task that undergoes systematic processing through various stages of planning and execution. In this setting, the user instructs the UGV with a prompt:"Navigate to the garage to check if the delivery truck is still here." This initiates a series of operations in which the HLP devises a preliminary plan to visit the garage. Consequently, the Low-Level Planner dispatches commands to the UGV to execute the navigate_to_garage service coupled with a spin_service. The scenario continues as the UGV is sequentially tasked with navigating to the cafe to fetch a water bottle and deliver to the lounge. The UGV adapts to this complex, multi-step operation by leveraging the updated KB to facilitate navigation and object interaction.

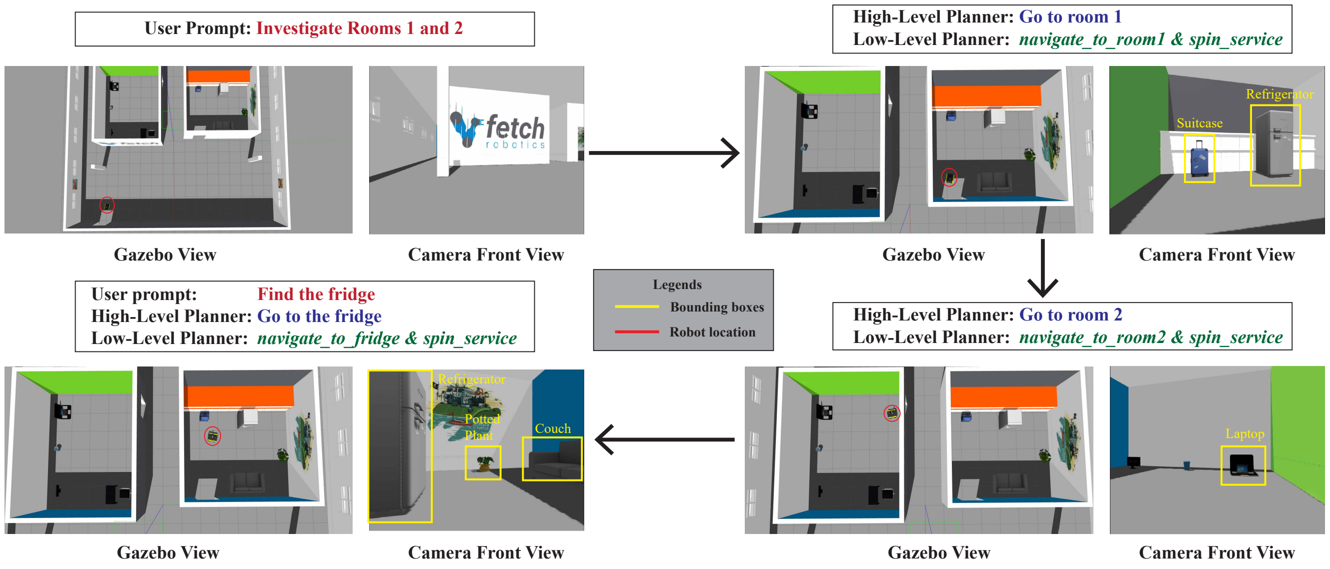
User Prompt: Investigate Rooms 1 and 2
High-Level Planner: Go to room 1
Low-Level Planner: navigate_to_room1 & spin_service
Gazebo View
Camera Front View
Gazebo View
Camera Front View
Suitcase
Refrigerator
User prompt: Find the fridge
High-Level Planner: Go to the fridge
Low-Level Planner: navigate_to_fridge & spin_service
Legends
Bounding boxes
Robot location
High-Level Planner: Go to room 2
Low-Level Planner: navigate_to_room2 & spin_service
Refrigerator
Potted Plant
Couch
Laptop
Gazebo View
Camera Front View
Gazebo View
Camera Front View

Figure 3. Sequential flow of the UGV's autonomous task execution, starting from the user prompt to the final KB update. The process is depicted in a residential setting, showcasing the UGV's interaction with the environment through the Gazebo and Rviz interfaces.

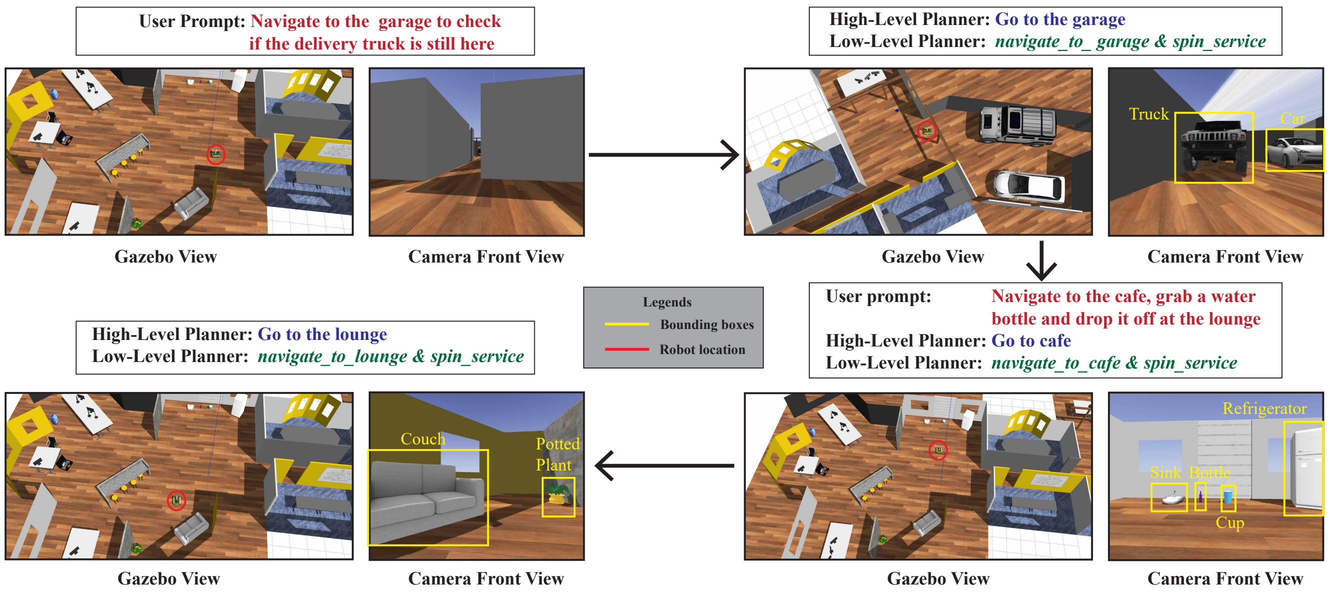
User Prompt: Navigate to the garage to check if the delivery truck is still here
High-Level Planner: Go to the garage
Low-Level Planner: navigate_to_ garage & spin_service
Gazebo View
Camera Front View
Truck
Car
Gazebo View
Camera Front View
Legends
Bounding boxes
Robot location
User prompt: Navigate to the cafe, grab a water bottle and drop it off at the lounge
High-Level Planner: Go to cafe
Low-Level Planner: navigate_to_cafe & spin_service
High-Level Planner: Go to the lounge
Low-Level Planner: navigate_to_lounge & spin_service
Couch
Potted Plant
Refrigerator
Sink
Bottle
Cup
Gazebo View
Camera Front View
Gazebo View
Camera Front View

Figure 4. Sequential flow of the UGV's autonomous task execution, starting from the user prompt to the final KB update. The process is depicted in a residential setting, showcasing the UGV's interaction with the environment through the Gazebo and Rviz interfaces.

4.2.3 Analysis of Task Handling with Adaptive Knowledge Base

The comparative performance of our ATLAS, equipped with a dynamic and growing KB, versus a system relying on a static, handcrafted KB, is outlined in Table 1. The table shows the success rate and efficiency of executing a range of navigation tasks of varying complexity defined by whether the landmark existed in the KB or not.

The tasks listed range from basic navigation to complex multi-step actions involving object interaction. Our system demonstrates its adaptability by successfully completing tasks that extend beyond the initial KB, showcasing an ability to assimilate new environmental data and utilize it effectively for task execution.

For simple navigation tasks, both systems perform equally well, as these tasks rely on pre-existing KB entries. However, ATLAS shows a distinct advantage for tasks that require interaction with objects not present in the initial KB.

Table 1. Task handling and success rate comparison between a static handcrafted Knowledge Base and our dynamic growing Knowledge Base for a residential setting (Environment 1).

Environment 1					
Complexity	**User Prompt**	**Handcrafted Knowledge-Base**		**Growing Knowledge-Base (Ours)**	
		No. of sub-tasks (success / total)	**Success**	**No. of sub-tasks (success / total)**	**Success**
Exists in Knowledge Base	*Navigate to Room 1*	*1/1*	*Yes*	*1/1*	*Yes*
	Navigate to Room 2	*1/1*	*Yes*	*1/1*	*Yes*
Added to the Knowledge Base Over time	*Navigate to the refrigerator, grab a beverage and deliver it at the laptop*	*0/2*	*No*	*2/2*	*Yes*
	Navigate to room 2, grab the cup and water the potted plant	*1/3*	*No*	*3/3*	*Yes*
	Turn off the TV in room 2 and head over the couch	*1/3*	*No*	*3/3*	*Yes*

Table 2. Task handling and success rate comparison between a static handcrafted Knowledge Base and our dynamic growing Knowledge Base for a warehouse setting (Environment 2).

Environment 2					
Complexity	**User Prompt**	**Handcrafted Knowledge-Base**		**Growing Knowledge-Base (Ours)**	
		No. of steps (success / total)	**Success**	**No. of steps (success / total)**	**Success**
Exists in Knowledge Base	*Navigate to the garage*	*1/1*	*Yes*	*1/1*	*Yes*
	Navigate to the lounge	*1/1*	*Yes*	*1/1*	*Yes*
	Navigate to the Cafe	*1/1*	*Yes*	*1/1*	*Yes*
	Navigate to the game room	*1/1*	*Yes*	*1/1*	*Yes*
Added to the Knowledge Base Over time	*Navigate to the garage to check if the delivery truck is still here*	*1/1*	*Yes*	*1/1*	*Yes*
	Navigate to the cafe, grab a water bottle and drop it off at the lounge	*1/3*	*No*	*3/3*	*Yes*
	Fill the cup in the cafe with water at the sink And water the potted plant	*1/4*	*No*	*4/4*	*Yes*

It successfully completes these tasks by dynamically updating its KB and effectively parsing complex user prompts into actionable sub-tasks, as indicated by the 'Number of steps' column.

The 'Growing Knowledge-Base (Ours)' column shows the system's ability to update the KB on the fly. For example, upon receiving the prompt to "Navigate to the refrigerator, grab a beverage, and deliver it to the laptop," ATLASnot only identified the refrigerator and laptop as useful landmarks but also updated its KB with these entities, thus allowing it to successfully execute the full task sequence in future attempts.

The success rate improvements from 'No' in the handcrafted system to 'Yes' in ours underline the advantages of autonomous learning. Tasks that were initially beyond the system's capability due to KB constraints become feasible as it learns, demonstrating the potential for continuous improvement and the capacity for handling increasingly complex scenarios.

This comparative analysis underscores the significance of a growing KB in advancing the capabilities of autonomous systems. It provides a clear illustration of how a dynamic and learning-based approach can yield superior performance, particularly in environments where adaptability and learning from interaction are essential.

Following the evaluation presented in Table 1, a similar analysis in a different context, Environment 2, a warehouse setting, further serves to show the efficacy of our adaptive KB system. Table 2 presents the system's performance across a range of tasks within the warehouse.

Again, the tasks vary from elementary single-step navigational tasks to multi-step operations. As shown, our approach surpasses the static KB system by achieving a 100% success rate, even for tasks that introduce new landmarks during the UGV's explorations.

For single-step navigation commands involving known

Table 3. Performance of ATLAS in semantic goal-reaching tasks where the target object is not directly listed in the Knowledge Base.

Complexity	**Prompt**	**Landmark (suggested / actual)**	**Success Rate**
Does not exist in KB	*What is the closest landmark near which i can find a sink.*	*Refrigerator / Refrigerator* Couch / Refrigerator	*70%*
	What is the closest landmark near which i can find a keyboard and mouse.	*Laptop / Laptop*	*100%*
	What is the closest landmark near which i can find a oven.	*Refrigerator / Refrigerator* Chair / Refrigerator	*80%*
	What is the closest landmark near which i can find a cloth hamper.	*Suitcase / Suitcase* Store / Suitcase	*60%*
	What is the closest landmark near which i can find a monitor.	*Laptop / Laptop*	*100%*

locations, such as ”Navigate to the game room” or ”Navigate to the lounge,” both systems show success, confirming the initial robustness of the static KB. However, in scenarios that involve new landmarks or require complex interactions, for instance, when instructed to ”Navigate to the cafe, grab a water bottle and drop it off at the lounge,” our system dynamically uses the previously added ’bottle’ from the KB, enabling successful task completion.

4.2.4 Semantic Goal Reaching

ATLAS’s capability extends beyond the retrieval of known landmarks within the KB. It can also perform advanced semantic reasoning by navigating to the most proximate and contextually relevant landmark when tasked to find an object absent from the KB. Table 3 illustrates the ATLAS’s performance on semantic goal-reaching tasks.

In scenarios where the user poses queries like ”What is the closest landmark near which I can find a sink,” the system can intelligently suggest ’Refrigerator’ as the landmark, leveraging the semantic clustering of objects in the environment. Despite ’Sink’ not being a direct entry in the KB, the LLM deduces that a sink can often be found in proximity to refrigerators in residential layouts.

The ’Landmark (suggested / actual)’ column showcases the LLM-prompted landmark versus the actual landmark. The success rate measures the number of times the appropriate landmark is prompted while the experiment is run 10 times.

This ability to perform semantic goal-reaching by connecting the dots between related objects significantly enhances ATLAS’s navigational efficiency, particularly in environments rich with contextually grouped entities. This ability makes the UGV not just a follower of direct commands but an intelligent agent capable of making inferences about its surroundings.

5. Conclusion

In this paper we introduced ATLAS an autonomous navigation system that integrates path planning, object detection and LLMs to navigate using natural language instructions. The key contributions include ATLAS’s ability to dynamically expand its knowledge base with new landmarks and use them to execute complex multi-step navigation tasks. Our results demonstrate ATLAS’s ability to adapt and learn incrementally achieving a high success rate across both known and semantically inferred navigation tasks in simulated environments.

Future work will explore extending ATLAS with more complex object detectors with a strong open-vocabulary detection capability and grounding ability. This allows us to further bridge the gap between the way humans and robots perceive and interact with their surroundings making autonomous navigation more versatile and intelligent.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022. 3

[2] 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. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018. 1, 2

[3] Matthew Chang, Theophile Gervet, Mukul Khanna, Sriram Yenamandra, Dhruv Shah, So Yeon Min, Kavit Shah, Chris Paxton, Saurabh Gupta, Dhruv Batra, et al. Goat: Go to any thing. *arXiv preprint arXiv:2311.06430*, 2023. 3

[4] Howard Chen, Alane Suhr, Dipendra Misra, Noah Snavely, and Yoav Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *Pro-*

ceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 12538–12547, 2019. 1

[5] Cohere. Command language model api, 2022. 1

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 1

[7] Russell Epstein. The cortical basis of visual scene processing. *Visual Cognition*, 12(6):954–978, 2005. 1

[8] 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. 3

[9] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, pages 9118–9147. PMLR, 2022. 3

[10] Ray Jackendoff and Barbara Landau. Spatial language and spatial cognition. In *Bridges between psychology and linguistics*, pages 157–182. Psychology Press, 2013. 1

[11] Glenn Jocher. YOLOv5 by Ultralytics, 2020. 3

[12] Boyi Li, Philipp Wu, Pieter Abbeel, and Jitendra Malik. Interactive task planning with language models. *arXiv preprint arXiv:2310.10645*, 2023. 3

[13] Yujie Lu, Weixi Feng, Wanrong Zhu, Wenda Xu, Xin Eric Wang, Miguel Eckstein, and William Yang Wang. Neuro-symbolic procedural planning with commonsense prompting. *arXiv preprint arXiv:2206.02928*, 2022. 3

[14] Arjun Majumdar, Ayush Shrivastava, Stefan Lee, Peter Anderson, Devi Parikh, and Dhruv Batra. Improving vision-and-language navigation with image-text pairs from the web. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*, pages 259–274. Springer, 2020. 2

[15] Hasib-Al Rashid and Tinoosh Mohsenin. Hac-pocd: Hardware-aware compressed activity monitoring and fall detector edge poc devices. In *2023 IEEE Biomedical Circuits and Systems Conference (BioCAS)*, pages 1–5. IEEE, 2023. 2

[16] Hasib-Al Rashid, Pretom Roy Ovi, Aryya Busart, Carl Gangopadhyay, and Tinoosh Mohsenin. Tinym2net: A flexible system algorithm co-designed multimodal learning framework for tiny devices. *ArXiv*, 2022.

[17] Hasib-Al Rashid, Argha Sarkar, et al. Tinyvqa: Compact multimodal deep neural network for visual question answering on resource-constrained hardware. *ArXiv*, 2024.

[18] Hasib-Al Rashid et al. Tinym^{2}net-v2: A compact low power software hardware architecture for $\underline{M}$ulti$\underline{m}$odal deep neural networks. *ACM Transactions on Embedded Computing Systems*, 2023. 2

[19] Kai-Florian Richter and Stephan Winter. Landmarks. *Springer Cham Heidelberg New York Dordrecht London. doi*, 10:978–3, 2014. 1

[20] 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. 3

[21] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 11523–11530. IEEE, 2023. 3

[22] Jesse Thomason, Aishwarya Padmakumar, Jivko Sinapov, Nick Walker, Yuqian Jiang, Harel Yedidsion, Justin Hart, Peter Stone, and Raymond J Mooney. Improving grounded natural language understanding through human-robot dialog. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6934–6941. IEEE, 2019. 1

[23] Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *arXiv preprint arXiv:2305.16291*, 2023. 3

[24] Zhong-Qiu Zhao, Peng Zheng, Shou-tao Xu, and Xindong Wu. Object detection with deep learning: A review. *IEEE transactions on neural networks and learning systems*, 30 (11):3212–3232, 2019. 2

[25] Kaizhi Zheng, Kaiwen Zhou, Jing Gu, Yue Fan, Jialu Wang, Zonglin Di, Xuehai He, and Xin Eric Wang. Jarvis: A neuro-symbolic commonsense reasoning framework for conversational embodied agents. *arXiv preprint arXiv:2208.13266*, 2022. 3