CogExplore: Language Model-Guided Planning Informed by Prior States

by

Patrick Cooper

B.S., University of New Mexico, 2021

B.F.A., University of New Mexico, 2021

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Computer Science

2024

Committee Members:

Christoffer Heckman, Chair

Professor Bradley Hayes

Professor Nikolaus Correll

Cooper, Patrick (M.S., Computer Science)

CogExplore: Language Model-Guided Planning Informed by Prior States

Thesis directed by Prof. Christoffer Heckman

The integration of language models into robotics enhances executive decision-making capabilities and establishes a foundation for world model grounding, opening new avenues in robotic navigation. Advanced models like GPT, Mistral and Gemini enable robots to make well-informed decisions by processing data from Visual Question Answering (VQA) models and real-time object detectors, leveraging their capabilities for cutting-edge exploration. Moreover, these models can compress observations and actions into natural language, forming a memory that guides future navigation. We propose a comprehensive pipeline that employs a ROS-Unreal interface to evaluate a language model-driven frontier-point exploration strategy. This approach showcases intuitive reasoning, leverages past experiences for future navigation, and generates a human-readable record of the decision-making process.

Acknowledgements

Harel Biggie, Doncey Albin, Kristen Such, Ian Steagall, Micheal Lopez

Contents

Figures

Figure

Chapter 1

Introduction

Modern robotics heavily rely on the exploration of unmapped environments, crucial for applications such as search-and-rescue and assistive robotics. Most advanced algorithms today use hand-engineered heuristics like volumetric gain or preferred exploration headings. These strategies are effective in environments like caves as demonstrated by various studies [5, 37, 1], but often overlook the nuanced semantics and context of their surroundings [4]. Our research introduces a novel framework that employs natural-language-based representations to enhance robotic exploration. This approach uses semantic cues and historical data analysis in an office environment setting.

We utilize natural-language representations to navigate using context-rich features. This requires anchoring language elements to tangible world aspects the robot interacts with. Existing methods to link language with physical settings range from probabilistic graphs to learning-based models [34, 17, 21, 14]. Our model formulates this connection as a set of probabilistic planning and object points paired with linguistic priors. We then analyze a comprehensive dataset, including inputs from a Visual Question Answering (VQA) model, to determine optimal exploration paths. Specifically, our use of foundational models [7, 20] enhances our capability to contextualize and reason effectively.

Language Model (LM) guided navigation offers new prospects in robotic movement by integrating a sophisticated model of the world [4]. This approach not only considers geometric factors but also employs a broad understanding of environment-specific characteristics, like the likely locations of objects based on their typical settings. It also applies world knowledge systematically to

refine search strategies, utilizing tools like cameras more effectively and adapting exploration tactics as more information becomes available. This evolving strategy, which involves reassessing previous decisions—a principle akin to Bayesian updating—improves decision-making as exploration progresses.

However, LMs typically struggle with temporal reasoning due to their non-recurrent structure [38]. To combat this, we propose CogExplore, a strategy that simulates working memory to enhance navigation in semantic-rich spaces. Traditional navigation methods falter in tasks that require memory and sequential reasoning. CogExplore integrates a Visual Question Answering system and an object recognition framework to guide decisions based on accumulated knowledge. Our experimental setup in Unreal Engine and ROS thoroughly tests these capabilities in diverse environments.

Overall, our framework not only showcases the potential of LMs to adapt quickly to new settings but also highlights how nuanced understanding can direct ongoing exploration. This is particularly relevant in complex environments like homes needing assistive robotics or areas requiring urgent humanitarian aid.

Chapter 2

Related Works

Navigating with language-based representations requires grounding utterances to the physical environment. Various approaches have been used to associate language with the physical domain, ranging from probabilistic graph-based structures [34, 17, 21] to end-to-end learning-based methods [14]. Graph-based approaches have shown promising results but their generalizability is limited to a fixed training corpus. On the other hand, Large Language Models (LLMs) have proven to be adept at reasoning over large amounts of unsupervised training data [15, 35] using transformer-based backends [39]. These models, also known as foundation models due to their comprehensive knowledge, have recently been leveraged for task and motion planning using reinforcement learning on a set of skills [6], and in an end-to-end fashion [14] with object scene representation transformers [27]. While these approaches have made remarkable strides, they still suffer from the explainability problem and further, they don't share any of the path planning guarantees that traditional planning methods offer [25].

More recent work in robotic navigation has seen significant advancements with the incorporation of LLMs [41] and semantic cues. The approach of utilizing LLMs was first introduced as a heuristic planner constructed for manipulation tasks [2]. In [30], Shah et al. propose a novel approach that utilizes semantic guesswork from LLMs as heuristics for planning in unexplored environments. Unlike traditional methods that rely on geometric or direct instruction-based navigation, their methodology leverages the semantic understanding of LLMs to guide exploration indirectly, without requiring direct knowledge of the physical environment. This flexible approach highlights

a novel application of LLMs and departs from purely geometric or directly instructed navigation methods.

Our method relies on the presence of natural language descriptions of what is visible to the robot's cameras via multimodal models. The effectiveness of LLMs for reasoning over visual priors is discussed in the development of ViperGPT [33]. This system uses image patches returned by an object detector as data structures to solve reasoning tasks described via natural language, demonstrating the potential of LLMs to understand and reason about visual information.

Dai et al. present ORION (Open-woRld Interactive persOnalized Navigation) in [11], which integrates LLMs for personalized, interactive object navigation. ORION focuses on natural language interaction and personalization, proposing a task where robots navigate to user-specific objects in unseen environments through conversations with users. The framework comprises modules for perception, navigation, communication, and decision-making, showcasing the integration of LLMs for navigational reasoning and interactive user feedback.

Embodied evaluation of actions across objects with specific affordances in reference to natural language instruction has been explored by Saycan et al. in [28] using code generation. Language models contain sufficient world information to decompose basic tasks [18], but must be evaluated according to the metrics of executability and correctness. This work highlights the importance of assessing the practical applicability of LLM-generated instructions in real-world robotic scenarios.

Alternative approaches include the LAnguage Trajectory TransformEr (LATTE) [8] and CLIPort [32], which use end-to-end mechanisms to encode user intentions and translate directly to robot action. CLIPort encodes spatial and semantic attributes separately using individual U-Nets, combining them in the final layer to inform trajectory generation. Imitation learning, as seen in Hierarchical Universal Language Conditioned Policies (HULC) [24], uses language conditioning to construct controls for HULC using the semantic knowledge of VAPO, learning efficiently from unlabeled examples. These approaches demonstrate the potential of integrating language understanding with robotic control, enabling more intuitive and efficient robot behavior.

Prompt engineering plays a crucial role in tackling highly specific robotic navigation tasks.

In [40], Vemprala et al. introduce PromptCraft, a mechanism for handling the complexity inherent in the myriad ways of describing any one task in natural language. By carefully crafting prompts that elicit the desired behavior from LLMs, PromptCraft aims to bridge the gap between natural language instructions and executable robot actions.

Shah et al. present an approach called Language Frontier Guide (LFG) that leverages the semantic reasoning capabilities of LLMs to guide robotic exploration in novel environments [29]. Rather than directly following the narratives generated by LLMs, which can be ungrounded and arbitrarily wrong, LFG incorporates LLM inferences as heuristic scores into a search-based planning framework. By polling the LLM multiple times with task-relevant prompts and chain-of-thought reasoning, LFG empirically estimates the likelihood of subgoals leading to the goal query. These semantic "guesswork" scores are then combined with traditional geometric heuristics to bias the search towards promising frontiers. The authors demonstrate that this approach outperforms both uninformed exploration and other LLM-based methods on the challenging Habitat ObjectNav benchmark and in real-world environments, showcasing the potential of leveraging LLM-based semantic reasoning to enhance robotic navigation in unfamiliar settings.

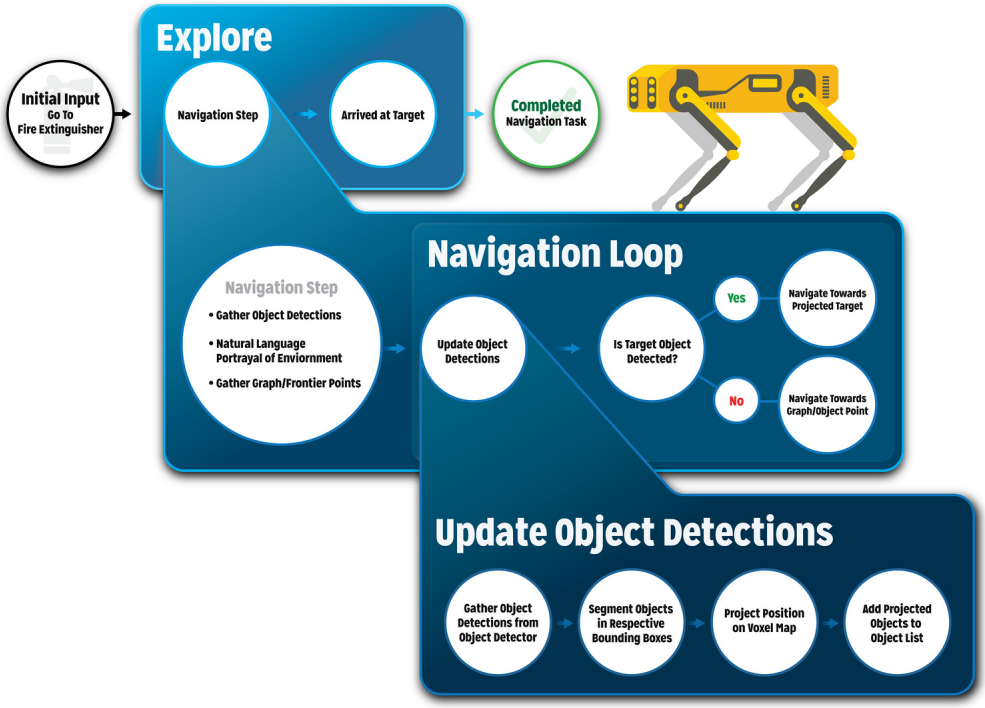
Chapter 3

Methods

Cog Explore, our exploration framework represents relevant features of the environment as natural language strings. Specifically, we represent possible states for the robot as $S = \{s_1, s_2, ..., s_n\}$ where s_i is a possible state for the robot. At each state the robot has a set of planning graph points $G = \{g_1, g_2, ..., g_k\}$, a set of object points $O = \{o_1, o_2, ..., o_m\}$ and a set of language priors $\mathcal{L}$. Graph points represent traversable areas of the environment and we represent them as list in text from $G_i = \{x, y, z\}$. Object points are represented in a similar fashion but with a corresponding label and probability obtained from an open vocabulary object detector [10] $O_i = \{x, y, z, c, p\}$ where c is the class and p is the confidence represented as a probability. Language priors L are obtained from a series of questions and answers asked by the agent, as shown in Figure 3.5. Questions are generated using a foundation model and answers are obtained using multimodal visual question and answering models [22]. In order to select the next best state we can model the task as a maximum likelihood estimate:

$$\arg \max_{s_1, s_2, ..., s_t} P(s_g | s_t, S, O, G, L) \prod_{i=1}^{t} P(s_i | s_{i-1}, S, O, G, L) \tag{3.1}$$

Figure 3.1: Overview of the CogExplore Method.



3.1 Design of Unreal Simulator

To ensure effective visual grounding for our method, we employ the Unreal Engine, a game engine renowned for its photorealism and advanced rendering capabilities. We develop a custom Unreal Simulator with a functional ROS bridge, enabling efficient simulation using our Spot digital twin constructed within the Unreal environment. The Spot Actor is equipped with three high-resolution cameras (1280x720), each with a 90-degree field of view (FOV), and a fully simulated lidar based on the Ouster lidar specifications.

One of the key strengths of the Unreal Engine is its ability to procedurally load and render massive amounts of geometry, allowing for the simulation of city-scale environments with unprecedented detail and fidelity. This capability is particularly relevant for robotics applications, as it enables the creation of complex, realistic environments in which to test and validate autonomous systems. By leveraging Unreal's advanced rendering pipeline, including techniques such as level-

Figure 3.2:
Spot robot in the simulated street environment.

of-detail (LOD) management and occlusion culling, we can efficiently simulate expansive urban landscapes while maintaining optimal performance.

In addition to its rendering prowess, the Unreal Engine also provides a robust framework for integrating AI and physics simulations. The MassAI plugin, developed by Epic Games, is a powerful tool for simulating large-scale crowds and traffic within the Unreal environment. MassAI utilizes an entity-component-system (ECS) architecture, which allows for the efficient management and simulation of thousands of agents in real-time. By incorporating MassAI into our Unreal Simulator, we can populate our virtual environments with realistic crowds and traffic, further enhancing the fidelity and immersion of the simulation.

The MassAI framework offers a range of features and functionalities that are particularly well-suited for robotics applications. It provides a flexible and scalable approach to agent behavior modeling, allowing for the creation of complex, emergent behaviors through the composition of

simple, reusable components. MassAI also integrates seamlessly with Unreal's navigation and pathfinding systems, enabling agents to navigate and interact with the environment in a realistic and efficient manner. Furthermore, MassAI supports the simulation of heterogeneous agent populations, allowing for the inclusion of pedestrians, vehicles, and other entities within the same simulation.

Another key advantage of using the Unreal Engine for robotics simulation is its ability to generate photorealistic sensor data. By leveraging Unreal's advanced rendering capabilities, we can generate synthetic camera images, depth maps, and other sensor modalities that closely resemble their real-world counterparts. This is achieved through the use of physically-based rendering (PBR) techniques, which accurately model the interaction of light with surfaces and materials. By training our perception algorithms on this high-fidelity synthetic data, we can improve their robustness and generalization capabilities, ultimately leading to better performance in real-world scenarios.

To facilitate the integration of our Unreal Simulator with existing robotics frameworks and tools, we implement a ROS bridge that enables seamless communication between the simulated environment and the robot's software stack. This bridge allows for the efficient passing of sensor data, control commands, and other messages between the simulation and the robot, enabling a tight feedback loop between perception, planning, and control. The ROS bridge also enables the use of standard ROS tools and visualization techniques, such as RViz and Gazebo, further enhancing the usability and interoperability of our simulation platform.

Efficient passing of textures within Unreal utilizing the ROS bridge involved several key techniques. First, render targets are set up for each camera component, allowing the captured images to be efficiently accessed and manipulated. The rendering process is performed asynchronously in a separate thread to minimize performance impact on the main game thread. To publish the camera data over ROS, the rendered images are copied from the GPU to the CPU using a custom render command enqueued on the render thread. This ensures efficient transfer of the image data without blocking the game thread. The copied image data is then converted from BGRA to RGB format and packed into a ROS `sensor_msgs/Image` message, which is published to the corresponding ROS topic for each camera.

Camera info messages (`sensor_msgs/CameraInfo`) are also published for each camera, providing essential calibration and metadata to the ROS system. These messages include the camera's intrinsic matrix, distortion coefficients, and other relevant parameters. Furthermore, the Spot Actor's pose and odometry data are published as ROS `nav_msgs/Odometry` messages. The actor's position, orientation, linear velocity, and angular velocity are captured and converted to the appropriate ROS coordinate system before being populated into the Odometry message and published to the designated topic.

The use of Unreal Engine for robotics simulation is a proven approach, with several notable examples in the literature. AirSim [31] is an open-source simulator for drones and ground vehicles built on top of the Unreal Engine, which has been widely used for research in autonomous systems. CARLA (Car Learning to Act) [13] is another popular open-source simulator for autonomous driving research, which leverages the Unreal Engine's rendering capabilities to generate realistic urban environments and sensor data. Nvidia Isaac Sim [23] is a powerful robotics simulation platform that utilizes the Unreal Engine to provide high-fidelity visual rendering and physics simulation. UnrealCV [26] is a plugin for the Unreal Engine that enables the generation of synthetic computer vision datasets, which has been used for various applications such as object detection, semantic segmentation, and visual navigation. Finally, RobCoG (Robot Control Generator) [16] is a framework for generating robot control software from high-level specifications, which utilizes the Unreal Engine for simulation and visualization.

A common theme across these simulators is the need for photorealistic real-time rendering, or in other words, bridging the sim-to-real gap, specifically in terms of vision. By leveraging the advanced rendering capabilities of the Unreal Engine, these simulators are able to generate synthetic data that closely resembles real-world imagery, enabling the development and testing of perception algorithms in a safe and controlled environment. This is particularly important for robotics applications, where the ability to generate diverse and realistic training data is critical for improving the robustness and generalization of learning-based methods.

3.1.1 Exploration Planner

A cornerstone of CogExplore's autonomous navigation is the utilization of an exploration-based planner designed to ground the language model in selecting reliable waypoints. To initiate the planning process, simulated lidar data from the Unreal environment is transmitted via the Unreal-ROS bridge to a voxel-based mapping method.

The resulting voxel-based map facilitates the operation of GBPlanner [12], an advanced graph-based exploration planner that was instrumental in securing the top position for Team CEREBUS in the 2017 DARPA Subterranean Challenge [36]. GBPlanner utilizes a bifurcated planning architecture that combines local and global planning stages.

3.1.1.1 Local Planner

The local planner, working within a constrained boundary D_L around the robot's current pose ξ_0, builds a local rapidly-exploring random graph $\mathbb{G}_L$ (Algorithm 1) to maximize exploration (information) gain g, ensuring efficient pathfinding while avoiding collisions and adhering to traversability constraints (Algorithm 2).

One of the most novel features in the GBPlanner's local planner is its calculation of exploration gain g for a given path. Given a path $\sigma_i \in \sum_L, i = 1 \dots n$ with a set of vertices along that path $v_j^i \in \sigma_i, j = 1 \dots m_i$, the exploration gain $\mathbf{g}(\sigma_i)$ for the path is calculated by first discovering what the discounted volumetric gain is for that path $\mathbf{VG}_{\sigma_i}$. The discounted volumetric gain for a path $\mathbf{VG}_{\sigma_i}$ is calculated by summing the volumetric gain $\mathbf{VG}$ multiplied by a weight function for each vertex in the path:

$$\mathbf{VG}_{\sigma_i} = \sum_{j=1}^{m_i} \mathbf{VG}(v_j^i) \cdot e^{-\gamma_D \cdot D(v_1^i, v_j^i)} \tag{3.2}$$

The weight function $e^{-\gamma_D \cdot D(v_1^i, v_j^i)}$ exponentially diminishes the value of *volumetric gain for a given vertex in a given path* $\mathbf{VG}(v_j^i)$ by a hand-crafted coefficient γ_D multiplied by the Euclidean distance from the given vertex to the vertex at the robot's current pose $\mathrm{D}(v_1^i, v_j^i)$. Once the volumetric gain

Algorithm 1 Algorithm to build local graph

function BUILDLOCALGRAPH(ξ_0)
  $\mathbb{V} \leftarrow \{\xi_0\};\ \mathbb{E} \leftarrow \emptyset$
  $\mathbb{G}_L = (\mathbb{V}, \mathbb{E})$
  $D_L \leftarrow$ SetLocalBound(ξ_0)
  while $N_{\mathbb{V}} \leq N_{\mathbb{V},max}$ **and** $N_{\mathbb{E}} \leq N_{\mathbb{E},max}$ **do**
    $\xi_{rand} \leftarrow$ SampleFree(D_L)
    $\xi_{nearest} \leftarrow$ NearestVertex($\mathbb{G}_L, \xi_{rand}$)
    if CollisionFree($\xi_{rand}, \xi_{nearest}$) **then**
      $\mathbb{V} \leftarrow \mathbb{V} \cup \{\xi_{rand}\};\ \mathbb{E} \leftarrow \mathbb{E} \cup \{\xi_{rand}, \xi_{nearest}\}$
      $\Xi_{near} \leftarrow$ NearestVertices($\mathbb{G}_L, \xi_{rand}, \delta$)
      for all $\xi_{near} \in \Xi_{near}$ **do**
        if CollisionFree(ξ_{rand}, ξ_{near}) **then**
          $\mathbb{E} \leftarrow \mathbb{E} \cup \{\xi_{rand}, \xi_{near}\}$
        end if
      end for
    end if
  end while
  return $\mathbb{G}_L = (\mathbb{V}, \mathbb{E})$
end function

Algorithm 2 Local Planner

$\xi_0 \leftarrow$ GetCurrentConfiguration()
$\mathbb{G}_L \leftarrow$ BuildLocalGraph(ξ_0)
$\Sigma_L \leftarrow$ GetDijkstraShortestPaths(G_L, ξ_0)
ComputeVolumetricGain($\mathbb{G}_L$)
$g_{\text{best}} \leftarrow 0$
$\sigma_{L,\text{best}} \leftarrow \emptyset$
for all $\sigma \in \Sigma_L$ **do**
  $g_\sigma \leftarrow$ ExplorationGain(σ)
  if $g_\sigma > g_{\text{best}}$ **then**
    $g_{\text{best}} \leftarrow g_\sigma;\ \sigma_{L,\text{best}} \leftarrow \sigma$
  end if
end for
$\sigma_{L,\text{best}} \leftarrow$ ImprovePath($\sigma_{L,\text{best}}$)
return $\sigma_{L,\text{best}}$

for a given path $\mathbf{VG}_{\sigma_i}$ is calculated, it is then multiplied by an additional weight function to finally compute the value of exploration gain $\mathbf{g}(\sigma_i)$:

$$\mathbf{g}(\sigma_i) = e^{-\gamma_S \cdot S(\sigma_i, \sigma_{exp})} \cdot \mathbf{VG}_{\sigma_i} \tag{3.3}$$

The weight function $e^{-\gamma_S \cdot S(\sigma_i, \sigma_{exp})}$ exponentially diminishes the total value of *volumetric gain*

for a given path $\mathbf{VG}_{\sigma_i}$ by a hand-crafted coefficient γ_S multiplied by the *similarity* of the given path σ_i to the current estimated exploration direction σ_{exp}, as shown by $S(\sigma_i, \sigma_{exp})$. This helps disfavor paths that diverge greatly from the current estimated exploration direction, therefore preventing the robot from continuously switching back and forth from corridors to only maximize volumetric gain. The similarity metric used in GBPlanner was developed using the Dynamic Time Warping (DTW) method presented in Abraham Bachrach's [3] 2013 PhD thesis (section 4.1.1).

3.1.1.2 Global Planner

When the local planner exhausts viable paths ($\sigma \in \sum_L$) or strategic expansion is needed, the global planner activates. This planner incrementally constructs a sparse global graph $\mathbb{G}_G$ using the current best local graph path $\sigma_{L,best}$ (Algorithm 3), directing the robot towards unexplored areas or back to safety, thus optimizing exploration and ensuring efficient homing (Algorithm 4).

Algorithm 3 Build a Global Graph by Adding Paths from Local Graph

```
function BuildGlobalGraph(𝔾_G, 𝔾_L, ξ_0)
    σ_{L,best} ← GetBestPath(𝔾_L)
    AddPathToGraph(𝔾_G, σ_{L,best})
    Σ_L ← GetDijkstraShortestPaths(𝔾_L, ξ_0)
    Σ_principal ← DoClustering(Σ_L)
    for all σ ∈ Σ_principal do
        AddPathToGraph(𝔾_G, σ)
    end for
    return 𝔾_G
end function
```

In practice, CogExplore uses a combination of the local and global graphs generated by GBPlanner, maximizing the number of waypoints a LLM can choose from.

3.1.2 Design of Explore Loop Cycle

The exploration loop cycle is a critical component of our methodology, orchestrating the interaction between the Language Model (LLM), the Unreal simulation environment, and the various perception and planning modules. Its operation enables the robot to autonomously navigate and explore its surroundings, guided by high-level natural language instructions.

Algorithm 4 Global Planner

$v_0 \leftarrow$ GetCurrentVertex()
$\Sigma_{G0} \leftarrow$ GetDijkstraShortestPaths(G_G, v_0)
$\Sigma_{GH} \leftarrow$ GetDijkstraShortestPaths(G_G, v_{home})
$\Sigma_{\text{Frontier}} \leftarrow$ GetPotentialFrontier(F)
$g_{\text{best}} \leftarrow 0$
$\sigma_{G,\text{best}} \leftarrow \emptyset$
for all $\sigma \in \Sigma_{\text{Frontier}}$ **do**
    $g_\sigma \leftarrow \text{GlobalExplorationGain}_G(\sigma)$
    if $g_\sigma > g_{\text{best}}$ **then**
        $g_{\text{best}} \leftarrow g_\sigma$; $\sigma_{G,\text{best}} \leftarrow \sigma$
    end if
end for
$\sigma_{G,\text{best}} \leftarrow$ ImprovePath($\sigma_{G,\text{best}}$)
return $\sigma_{G,\text{best}}$

Upon instantiation, the LLM receives a carefully crafted prompt containing its task, such as "Go to the fire extinguisher" or "Go to the coffee table," serving as the initial guiding principle for the robot's exploration. To gain a deeper understanding of its environment and the context of its task, the LLM generates a stack of 3-5 strategically designed questions that are passed to a Visual Question Answering (VQA) model [22]. Prompting the VQA model required careful precision to mitigate its propensity for hallucination, leading to extensive experimentation and the resolution of a nonstandard parameterization. The payload for the VQA model includes key parameters such as "n_predict" to ensure concise answers, "temperature" to control output randomness, "repeat_klast_n" and "repeat_penalty" to discourage repetition, "top_k" and "top_p" to balance diversity and quality, and "presence_penalty" to manage token repetition. Additionally, the prompt is appended with an instruction for the model to "Be concise. Be accurate," which empirically improved performance. By carefully tuning these parameters, we optimized the VQA model's performance for generating accurate and concise answers to the LLM's questions, enhancing the robot's ability to acquire relevant environmental information and effectively navigate and complete its assigned tasks.

Informed by this visual grounding and the objective of its navigation task, the LLM generates a set of objects to search for within the environment. This targeted object detection enables the

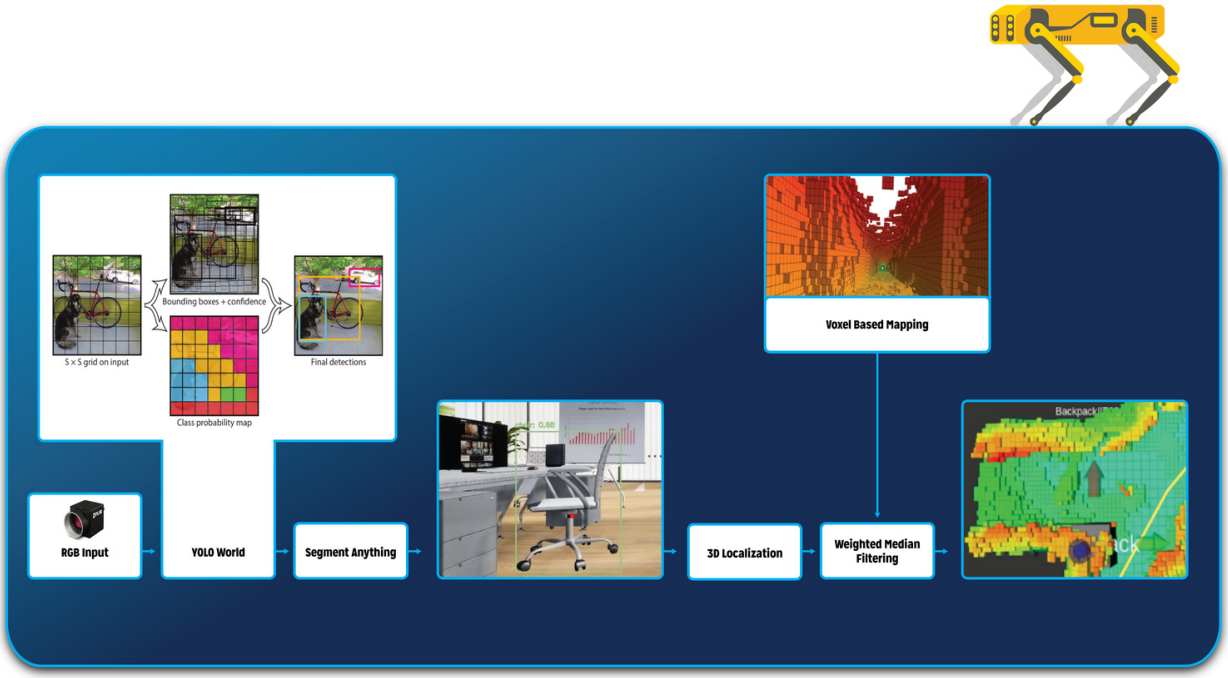
robot to focus its attention on elements that are likely to be relevant to its task. Once this set of object labels is sent, the object detection model, YOLO-World [10], begins processing the incoming stream of images. The object detection model employed in our methodology is based on an open-vocabulary architecture, allowing for flexible and adaptive recognition of objects in the environment.

Simultaneously, simulated lidar data from the Unreal environment is transmitted via the ROS bridge to a voxel-based mapping module. This mapping module integrates the lidar data over time, constructing a 3D occupancy map of the environment. The voxel-based representation provides a compact and efficient encoding of the spatial structure, enabling real-time processing and planning.

Within the LLM explore loop, the role of GBPlanner (detailed in previous subsection) extends to handling graph generation, path planning, and path-tracking. The most crucial application of GBPlanner is its graph construction of possible waypoints, including both local and frontier points, which are identified through the voxel-based mapping method. These points are then subsampled for uniformity, providing the LLM with a rich set of data points that represent the mapped space. The list of available graph points also includes object points corresponding to the detected objects in the environment. Informed by this information, the LLM deliberates on the next optimal point for navigation. Once a navigation point is selected, GBPlanner constructs the path and communicates it as a series of transforms to the digital twin, which then navigates to the designated location within the simulated Unreal scene.

If an object belonging to one of the specified classes is detected, an image segmentation model called "Segment Anything" operates on the detected image patch. This segmentation step is crucial for precise localization of the object within the image frame. The segmentation model identifies the centroid of the object, providing a robust and reliable reference point. This centroid is then used to inform the projection step, which maps the 2D image coordinates to the corresponding 3D point in the simulated environment. By leveraging the camera calibration parameters and depth information, the projection step yields an x, y, z coordinate in the simulation space corresponding to the detected and segmented object. These object coordinates are appended to a list of object

Figure 3.3: Explanation of Vision Projection Pipeline



points maintained by the system.

Armed with this rich set of information, including the subsampled graph points, object points, and its own internal representation of the environment and task, the LLM deliberates on the next optimal point for navigation. This decision-making process takes into account the broader navigation goals and the specific objectives of the current task. By considering the spatial layout, object locations, and prior knowledge, the LLM aims to select a point that maximizes the expected information gain and brings the robot closer to its target.

Once a navigation point is selected, GBPlanner constructs a path to reach that point. The path is communicated to the digital twin in the form of a series of transforms, specifying the desired trajectory. The digital twin, embodied as the Spot Actor within the Unreal simulation, then navigates to the designated location by following the provided path.

Figure 3.4: Example of Bounding Boxes and Corresponding Projection Points.

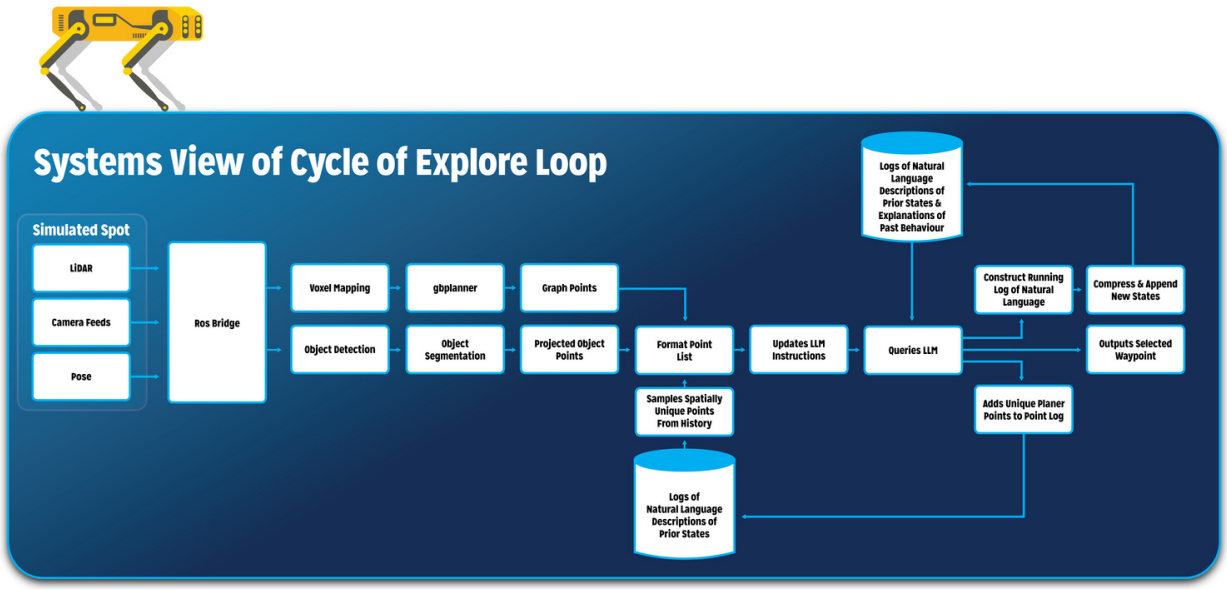
If the Spot Actor successfully reaches its destination, the LLM explore loop is re-initiated. This subsequent iteration incorporates the reached destination as an additional prior state, enriching the LLM's understanding of the environment and the progress made thus far. This iterative process allows for continuous refinement of the navigation strategy, adapting to new information and experiences gained during the exploration.

In certain scenarios, the exploration loop may be interrupted by the detection of an object of interest. If the target object specified in the initial prompt is detected during navigation, the system may override the planned path and compute a more direct route to the object. This adaptive behavior prioritizes reaching the target object efficiently, leveraging the real-time perception capabilities of the system.

If the Spot Actor successfully reaches the detected object's location, the simulation is considered complete, and a success signal is triggered. However, in cases where the Spot Actor arrives at the object's location without triggering the simulation's termination, a separate LLM prompting scheme is activated. This scheme is designed to handle unexpected or ambiguous situations encountered during the exploration process. It encodes the current state and the detected object's

information as an interruption, indicating an attempt to resolve the navigation task. This encoded interruption is then appended to the history of prior states maintained by the LLM. By incorporating these interruptions into its memory, the LLM gains the ability to reason about and adapt to unexpected events, enhancing its robustness and flexibility in handling complex exploration scenarios.

Figure 3.5: Overview of the CogExplore Explore Loop.



The explore loop cycle described above represents a sophisticated integration of perception, planning, and decision-making components, orchestrated by the LLM. By leveraging advanced techniques in computer vision, lidar-based mapping, graph-based planning, and natural language processing, our methodology enables autonomous exploration guided by high-level instructions. The iterative nature of the loop allows for continuous adaptation and refinement, while the ability to handle interruptions and unexpected events enhances the system's resilience in real-world scenarios. Through this intricate interplay of modules, our approach aims to push the boundaries of language-guided robot exploration, paving the way for more intelligent and adaptive autonomous systems.

3.1.3 Design of Prompting Scheme for LLM

The prompting scheme for the Language Model (LLM) in CogExplore is a critical component that enables the LLM to effectively guide the robot's exploration based on the current state of the environment and the robot's navigation history. The design of the prompting scheme involved an extensive iterative process, as LLMs are highly sensitive to the specific presentation of the input information [9]. To accommodate the diverse range of information required for effective exploration, we developed a highly modular prompt construction system.

The prompt engineering process began with identifying the key components necessary for the LLM to make informed decisions about waypoint selection and exploration strategies. These components included the task description, contextual information about the environment, available waypoints, navigation history, and specific instructions for the LLM.

To determine the optimal structure and content of each component, we conducted a series of experiments with different prompt variations. These experiments involved systematically modifying the prompts and evaluating the quality of the LLM's responses in terms of their relevance, coherence, and alignment with the desired exploration behaviors.

One of the key findings from these experiments was the importance of providing the LLM with a clear and concise task description. We observed that including the specific navigation task query at the beginning of the prompt, using the `*INSERT_QUERY_HERE*` placeholder, helped the LLM maintain focus on the overall objective throughout the exploration process.

Another critical aspect of the prompt engineering process was striking the right balance between providing sufficient contextual information and avoiding overloading the LLM with irrelevant details. We experimented with different levels of detail in the scene descriptions, object lists, and navigation history, and found that concise yet informative representations yielded the best results.

To encourage the LLM to prioritize the exploration of new and informative areas, we incorporated explicit guidance on the significance of frontier points and newly discovered regions. By emphasizing the importance of these points in the prompt, we observed a notable improvement in

the LLM's ability to make strategic waypoint selections.

Handling interruptions and unexpected situations during navigation was another key consideration in the prompt engineering process. We introduced a dedicated placeholder, `*INSERT_INTERRUPT_DESCRIPTION*`, to allow for the insertion of relevant information about such events. This enabled the LLM to adapt its guidance based on the specific circumstances encountered by the robot.

The inclusion of model-specific instructions, through the `*INSERT_SPECIFIC_MODEL_INSTRUCTIONS*` placeholder, allowed us to tailor the prompting scheme to the strengths and weaknesses of different LLMs. By providing targeted guidance on the desired output format and reasoning process, we were able to optimize the LLM's responses for the robot's navigation system.

To facilitate the integration of the LLM's outputs with the robot's decision-making process, we designed a structured output format in the form of a fillable template. This template included fields for the selected waypoint, a description of the environment, and the reasoning behind the chosen waypoint. We refined the template through iterative testing to ensure that it captured the necessary information while remaining easily parsable by the robot's navigation system.

The prompt engineering process also involved the development of a state compression mechanism, where the LLM's generated guidance was compressed into a concise summary of 50-100 words. This compression step aimed to capture the semantic richness of the environment at each point in time while maintaining a manageable size for the navigation history. We experimented with different compression strategies and evaluated their effectiveness in preserving the most relevant information for the LLM's decision-making process.

Throughout the prompt engineering process, we continuously evaluated the LLM's performance on a diverse set of navigation tasks and environments. This iterative evaluation allowed us to identify areas for improvement and refine the prompting scheme accordingly. We also solicited feedback from domain experts to ensure that the prompts aligned with best practices in autonomous exploration and robot navigation.

The resulting prompting scheme is a product of extensive experimentation, iterative refinement, and domain expertise. By carefully crafting each component of the prompt and optimizing their interplay, we have developed a robust and effective means of guiding the LLM's decision-making process for autonomous exploration in complex environments.

Chapter 4

Results

We examine our method across 3 variations of the same office space. This office space includes a meeting room, lounge area, and a hallway. We note that each of these is semantically distinct, intended to engage with the contextual reasoning of our method. Our results demonstrate the ability of LLM models to effectively reason over spatial, semantic, and prior states to efficiently solve navigation tasks. A qualitative characterization of the reasoning exhibits a high level of understanding of the environment and consistent grounded reasoning.

To demonstrate the effectiveness of our CogExplore method, we present a comprehensive analysis of a single run in the office environment, where the robot was tasked to "Go to the fire extinguisher" for the environment configuration Fire Extinguisher 1 for the environment Office Environment 1. This run showcases the key features of our approach, including the LLM's ability to reason about the environment, leverage past observations, guide the robot towards the target object, and adapt to unexpected situations.

At the start of the run (t=0.0s), the LLM is provided with a description of the domestic environment, characterized by large windows, wooden floors, and scattered furniture such as chairs. Despite the absence of a fire extinguisher in the initial observations, the LLM decides to explore an area near the chairs (point 83, x=-1.125, y=6.68, z=-0.08). The reasoning behind this decision is particularly noteworthy:

"By exploring this new area near the chairs, we might have a higher chance of locating the fire extinguisher, which is a critical safety equipment related to our navigation task. Additionally,

going to a new area with different objects can provide a fresh perspective for the robot's cameras and lidar to scan for potential targets."

This decision demonstrates the LLM's understanding of the semantic relationship between common areas where people gather (indicated by chairs) and the likely placement of safety equipment like fire extinguishers. It also shows the model's awareness of the need to explore new areas to gain fresh perspectives for the robot's sensors, highlighting the importance of the exploration-based planning approach employed by CogExplore.

As the run progresses (t=24.6s), the robot reaches the area near the chairs and observes a dining table, potted plants, and the absence of any signs of potential hazards or emergency equipment. At this point, the LLM is presented with a new set of 128 points to choose from, including both graph points representing frontier areas and object points detected by the perception system. The LLM carefully considers each point, reasoning about their potential to lead towards the target object based on the current environment description and past observations.

After analyzing the available options, the LLM selects point 108, labeled as an "Object fire extinguisher" with coordinates x=-4.275, y=14.77, z=0.38 and a distance of 9.23 meters from the robot's current position. The model provides the following rationale for its decision:

"Given the information provided in the past states, I have explored areas with chairs and potted plants, but have not yet located the fire extinguisher. The fire extinguisher is crucial for emergency situations, and as it is an object that stands out in terms of its shape and color, it is likely to be placed in a strategic and easily accessible location."

Here, the LLM demonstrates its ability to leverage past observations to inform its decision-making. It recognizes that despite exploring areas with chairs and potted plants, the target object has not been found. The model also reasons about the distinctive shape and color of fire extinguishers and their strategic placement in easily accessible locations, further guiding the robot towards the target. This showcases the effectiveness of the iterative nature of the explore loop cycle, which allows the LLM to continuously refine its navigation strategy based on new information and experiences gained during the exploration.

As the robot navigates towards the selected fire extinguisher point, an unexpected interruption occurs at t=30.7s. The robot is forced to stop at the closest plannable point (x=-3.921, y=14.235, z=0.047) due to an obstacle or other environmental constraint. This interruption suggests that the object detection may have been inaccurate, as the target object was not found at the expected location.

Despite this setback, the LLM exhibits robustness and adaptability. It encodes the interruption and appends it to the memory of prior states, allowing the model to reason about and learn from the unexpected event. By incorporating this information into its decision-making process, the LLM demonstrates its ability to handle ambiguous situations and adjust its navigation strategy accordingly.

Throughout the run, the LLM consistently reasons about the spatial layout of the environment, the semantic relationships between objects, and the need to explore new areas to find the target object. It leverages past observations to inform its decisions and exhibits robustness to perceptual uncertainty. These capabilities, enabled by the LLM's vast knowledge and reasoning skills, contribute to the effectiveness of CogExplore in guiding the robot towards the target object in previously unseen environments.

Based on our description here the success of this run can be attributed to several key components of the CogExplore methodology:

(1) The careful design of the prompting scheme, which allows the LLM to incorporate custom visual grounding information, prior states, interruptions, graph and object points, and specific instructions tailored to the strengths and weaknesses of the model. This modular construction of the prompt enables the LLM to reason effectively about the environment and adapt to new information.

(2) The integration of advanced perception techniques, such as the open-vocabulary object detection model and the Segment Anything model for precise object localization. These components provide the LLM with a comprehensive understanding of its surroundings,

enabling it to make informed decisions about navigation.

(3) The iterative nature of the explore loop cycle, which allows for continuous refinement of the navigation strategy based on new information and experiences gained during the exploration. This adaptive approach enables the LLM to handle unexpected events and maintain focus on the target object. The ability to handle interruptions and ambiguous situations through a separate LLM prompting scheme. By encoding these interruptions into the LLM's memory of prior states, CogExplore enhances the system's robustness and flexibility in handling complex exploration scenarios.

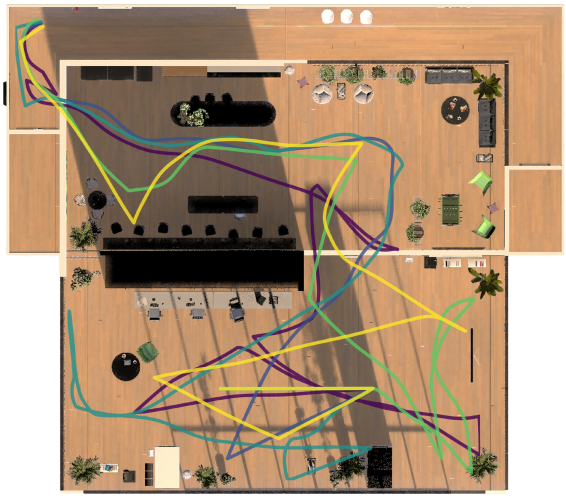
This run provides a detailed illustration of how CogExplore leverages the reasoning capabilities of LLMs to guide robotic exploration in unknown environments. By carefully considering the available information, including environment descriptions, object detections, and past experiences, the LLM makes informed decisions to navigate towards the target object efficiently.

The run highlights the LLM's ability to reason about the semantic relationships between objects and their likely locations, prioritize the exploration of new areas, and adapt to unexpected situations. These capabilities, combined with the advanced perception techniques, graph-based planning, and the iterative explore loop cycle, enable the robot to make steady progress towards the target object despite the challenges encountered along the way. By leveraging the vast knowledge and reasoning skills of LLMs, CogExplore enables more intelligent and adaptive autonomous exploration in complex, real-world environments.

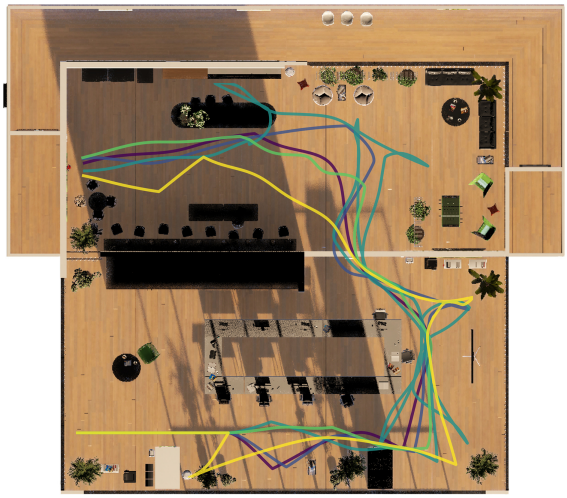
Despite these encouraging results, a notable weakness can be seen across these trajectories where the spot actor will overshoot the target and later perform an interrupt while en route to double back to the target. Overshooting is due to the inference time required by the vision models and represents a limitation of our method.

We release both our environments and their corresponding parameterization (ie robot start position and target location) for particular experiments alongside our project's code release.

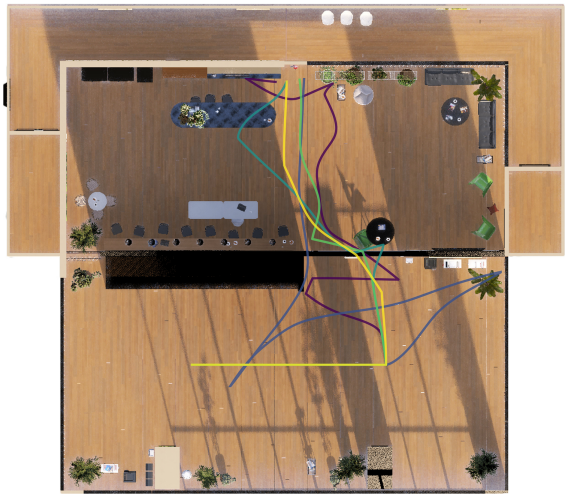
Figure 4.1: Trajectories for reaching Fire Extinguishers in Office Environment 1.



(a) Trajectory for reaching Fire Extinguisher 1



(b) Trajectory for reaching Fire Extinguisher 2



(c) Trajectory for reaching Fire Extinguisher 3

Chapter 5

Discussion

This study revealed distinct "personalities" among the language models (LMs) in their approach to exploratory behaviors. GPT-3.5-Turbo relied heavily on object detectors, often conducting cursory environment surveys when searching for objects. In contrast, Mistral displayed a more cautious personality, meticulously probing high-potential areas before proceeding. Notably, each model consistently adhered to its chosen strategy throughout an entire run.

Our findings demonstrate that LMs can successfully augment existing robotic planner capabilities, such as those of gbplanner. By leveraging spatial intuitions and logs of prior states, LMs provide semantic reasoning and bias behavior towards specific objectives, effectively guiding tactical exploration. Despite not being explicitly trained for these tasks, the models exhibited the ability to reason over prior states, leading to the emergence of efficient navigation capabilities, including peaking, sweeping, and poking behaviors. These behaviors suggest that the robot navigates with an understanding of how space can be observed.

However, a significant limitation lies in the grounding behavior enabled by vision models, particularly object detection and visual question answering (VQA). VQA models are highly susceptible to leading questions from the LM, and object detection models occasionally hallucinate target objects, resulting in false positives and incorrect detections. To mitigate these issues, careful configuration and tuning of the simulation to closely match the photoreal training distribution proved critical. Additionally, we employed prompt engineering to instruct the model to avoid relying on any single observation of its environment. Remarkably, the LM demonstrated robustness

against hallucinations, as exemplified by the following excerpt from office environment 1, "go to the fire extinguisher 2", utilizing GPT-3.5:

> "I recommend selecting point number 98...
>
> I suggest choosing this object point because it aligns with the navigation task of finding the fire extinguisher. Despite the label as a fire extinguisher, previous points visited were not successful in locating it, indicating potential mislabeling. However, considering the proximity of this object point and the importance of the fire extinguisher for safety measures in a domestic setting, it is a strategic choice to explore this new area and potentially discover the target object."

In this example, the LM successfully determines that previous detections of the target were false and selects a new promising occurrence of the target. This skepticism about observations prevents a vicious cycle where the model becomes disoriented by spurious object detections and other grounding information.

The LM models exhibit trust in their camera systems, as evidenced by their exploration behavior. The systems survey areas before proceeding to the next area, demonstrating the LM's understanding of the importance of exploring one area sequentially before moving on to the next. This behavior suggests both a spatial inference that points can compose distinct rooms and an acknowledgment that areas should be explored sequentially and progressively. An excerpt from the LM's reasoning for office environment 1, "go to the fire extinguisher", illustrates this:

> "I need to select a waypoint from a numbered list of graph points...
>
> My environment can be described as domestic, with large windows, a kitchen area, furniture, and a potted plant that requires periodic maintenance. There are no red objects in the environment, and there are no signs of potential hazards present.
>
> The reason I chose point 17 is that it is a newly discovered area with coordinates x=-2.68, y=9.76, z=-0.04. This point is at a distance of 7.29 meters, which is within reach but far enough to explore a different vantage point within the domestic setting. By exploring this new area, I aim to gain fresh perspectives and potentially uncover the location of the fire extinguisher, leveraging the robot's cameras and lidar capabilities.
>
> This decision aligns with the objective of efficient navigation towards the fire extinguisher by exploring uncharted territories and avoiding revisiting areas already explored."

In this reasoning, the model weighs the benefits of remaining within its current region, given the absence of semantically relevant elements like "red objects" or "potential hazards." Consequently, it elects to travel to a nearby discovered region, demonstrating both a spatial understanding of the environment and an appreciation of the semantic inputs gathered via its VQA interface.

LMs can generate explanations that constrain the set of possible behaviors, even if the explanations are factually incorrect. Given a set of explanations A for a behavior B, the set of behaviors consistent with A is a proper subset of all possible behaviors ($B_A \subset B$). In our experiments, all generated explanations were relevant to their associated behaviors, even when the behavior-explanation pairs were based on hallucinated perceptions, demonstrating the power of explanations to constrain behavior.

We note that successful fine-tuning of VQA models on specific datasets has been effectively demonstrated [19]. Future work can focus on fine-tuning existing VQA models to achieve superior results in particular target navigation environments, such as urban or domestic settings.

Chapter 6

Conclusion

Our work demonstrates the potential for enhancing autonomous navigation performance by augmenting robotic planners with grounding from vision models and strategic reasoning provided by large language models (LLMs) as embodied agents. LLMs exhibit the ability to effectively reason using spatial, semantic, and temporal information, leading to the emergence of strategic navigation behaviors such as peaking, sweeping, and poking. Despite limitations in the grounding behavior enabled by vision models, particularly object detection and visual question answering (VQA), careful configuration and prompt engineering prove critical in addressing these challenges. The LMs' ability to reason about the importance of exploring areas sequentially and progressively, while weighing the benefits of remaining within the current region against exploring uncharted territories, demonstrates their spatial understanding and appreciation of semantic inputs. The power of explanations to constrain behavior is evident, as generated explanations remain relevant to their associated behaviors, even when based on hallucinated perceptions. Future work should focus on fine-tuning VQA models for specific navigation environments and exploring the relationships between LMs' reasoning, the quality of grounding information, and the resulting navigation behaviors to develop more robust and efficient autonomous navigation systems.

Bibliography

[1] Ali Agha, Kyohei Otsu, Benjamin Morrell, David D Fan, Rohan Thakker, Angel Santamaria-Navarro, Sung-Kyun Kim, Amanda Bouman, Xianmei Lei, Jeffrey Edlund, et al. Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge. arXiv preprint arXiv:2103.11470, 2021.

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, ..., and Yuke Zhu. Do as i can, not as i say: Grounding language in robotic affordances, 2022.

[3] Abraham Bachrach. Trajectory bundle estimation for perception-driven planning. 2013.

[4] Harel Biggie, Ajay Narasimha Mopidevi, Dusty Woods, and Christoffer Heckman. Tell Me Where to Go: A Composable Framework for Context-Aware Embodied Robot Navigation. In Conference on Robot Learning. PMLR, 11 2023.

[5] Harel Biggie, Eugene R Rush, Danny G Riley, Shakeeb Ahmad, Michael T Ohradzansky, Kyle Harlow, Michael J Miles, Daniel Torres, Steve McGuire, Eric W Frew, et al. Flexible supervised autonomy for exploration in subterranean environments. Field Robotics, 3:125–189, 2023.

[6] Anthony Brohan, Yevgen Chebotar, Chelsea Finn, Karol Hausman, Alexander Herzog, Daniel Ho, Julian Ibarz, Alex Irpan, Eric Jang, Ryan Julian, et al. Do as i can, not as i say: Grounding language in robotic affordances. In Conference on Robot Learning, pages 287–318. PMLR, 2023.

[7] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. CoRR, abs/2005.14165, 2020.

[8] Anthony Bucker, Lucas Figueredo, Sami Haddadin, Ashish Kapoor, Shih-Yun Ma, Sai Vemprala, and Rogerio Bonatti. Latte: Language trajectory transformer, 2022.

[9] Yanda Chen, Ruiqi Zhong, Narutatsu Ri, Chen Zhao, He He, Jacob Steinhardt, Zhou Yu, and Kathleen McKeown. Do models explain themselves? counterfactual simulatability of natural language explanations, 2023.

[10] Tianheng Cheng, Lin Song, Yixiao Ge, Wenyu Liu, Xinggang Wang, and Ying Shan. Yolo-world: Real-time open-vocabulary object detection. In Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR), 2024.

[11] Yinpei Dai, Run Peng, Sikai Li, and Joyce Chai. Think, act, and ask: Open-world interactive personalized robot navigation. ICRA, 2024.

[12] Tung Dang, Marco Tranzatto, Shehryar Khattak, Frank Mascarich, Kostas Alexis, and Marco Hutter. Graph-based subterranean exploration path planning using aerial and legged robots. Journal of Field Robotics, 10 2020.

[13] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In Proceedings of the 1st Annual Conference on Robot Learning, pages 1–16, 2017.

[14] Danny Driess, Fei Xia, Mehdi SM Sajjadi, Corey Lynch, Aakanksha Chowdhery, Brian Ichter, Ayzaan Wahid, Jonathan Tompson, Quan Vuong, Tianhe Yu, et al. Palm-e: An embodied multimodal language model. arXiv preprint arXiv:2303.03378, 2023.

[15] Luciano Floridi and Massimo Chiriatti. Gpt-3: Its nature, scope, limits, and consequences. Minds and Machines, 30:681–694, 2020.

[16] Andrei Haidu and Michael Beetz. Automated models of human everyday activity based on game and virtual reality technology. In 2019 International Conference on Robotics and Automation (ICRA), pages 2606–2612, 2019.

[17] Thomas M Howard, Stefanie Tellex, and Nicholas Roy. A natural language planner interface for mobile manipulators. In 2014 IEEE International Conference on Robotics and Automation (ICRA), pages 6652–6659. IEEE, 2014.

[18] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents, 2022.

[19] Ashhadul Islam, Md. Rafiul Biswas, Wajdi Zaghouani, Samir Brahim Belhaouari, and Zubair Shah. Pushing boundaries: Exploring zero shot object classification with large multimodal models. In 2023 Tenth International Conference on Social Networks Analysis, Management and Security (SNAMS), pages 1–6. IEEE, 2023.

[20] Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mixtral of experts, 2024.

[21] Thomas Kollar, Stefanie Tellex, Deb Roy, and Nicholas Roy. Toward understanding natural language directions. In 2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI), pages 259–266. IEEE, 2010.

[22] Haotian Liu, Chunyuan Li, Yuheng Li, Bo Li, Yuanhan Zhang, Sheng Shen, and Yong Jae Lee. Llava-next: Improved reasoning, ocr, and world knowledge, January 2024.

[23] Viktor Makoviychuk, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, and Gavriel State. Isaac gym: High performance GPU based physics simulation for robot learning. In Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2), 2021.

[24] Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned robotic imitation learning over unstructured data. IEEE Robotics and Automation Letters (RA-L), 7(4):11205–11212, 2022.

[25] Iram Noreen, Amna Khan, and Zulfiqar Habib. Optimal path planning using rrt* based approaches: a survey and future directions. International Journal of Advanced Computer Science and Applications, 7(11), 2016.

[26] Weichao Qiu, Fangwei Zhong, Yi Zhang, Siyuan Qiao, Zihao Xiao, Tae Soo Kim, Yizhou Wang, and Alan Yuille. Unrealcv: Virtual worlds for computer vision. ACM Multimedia Open Source Software Competition, 2017.

[27] Mehdi SM Sajjadi, Daniel Duckworth, Aravindh Mahendran, Sjoerd van Steenkiste, Filip Pavetic, Mario Lucic, Leonidas J Guibas, Klaus Greff, and Thomas Kipf. Object scene representation transformer. Advances in Neural Information Processing Systems, 35:9512–9524, 2022.

[28] Ozan Saycan, Shivam Garg, Elliot Meyerson, Jonathan Tsai, Vicente Ordonez, Roozbeh Mottaghi, and Ali Farhadi. Progprompt: Generating situated robot task plans using large language models, 2022.

[29] Dhruv Shah, Michael Equi, Blazej Osinski, Fei Xia, Brian Ichter, and Sergey Levine. Navigation with large language models: Semantic guesswork as a heuristic for planning, 2023.

[30] Dhruv Shah, Aakash Malhotra, Sumegh Singh, and Abhinav Gupta. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action, 2021.

[31] Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In Field and Service Robotics, 2017.

[32] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In Proceedings of the 5th Conference on Robot Learning (CoRL), 2021.

[33] Dídac Surís, Sachit Menon, and Carl Vondrick. Vipergpt: Visual inference via python execution for reasoning. Proceedings of IEEE International Conference on Computer Vision (ICCV), 2023.

[34] Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In Proceedings of the AAAI Conference on Artificial Intelligence, volume 25, pages 1507–1514, 2011.

[35] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971, 2023.

[36] Marco Tranzatto, Mihir Dharmadhikari, Lukas Bernreiter, Marco Camurri, Shehryar Khattak, Frank Mascarich, Patrick Pfreundschuh, David Wisth, Samuel Zimmermann, Mihir Kulkarni, Victor Reijgwart, Benoit Casseau, Timon Homberger, Paolo De Petris, Lionel Ott, Wayne Tubby, Gabriel Waibel, Huan Nguyen, Cesar Cadena, Russell Buchanan, Lorenz Wellhausen, Nikhil Khedekar, Olov Andersson, Lintong Zhang, Takahiro Miki, Tung Dang, Matias Mattamala, Markus Montenegro, Konrad Meyer, Xiangyu Wu, Adrien Briod, Mark Mueller, Maurice Fallon, Roland Siegwart, Marco Hutter, and Kostas Alexis. Team cerberus wins the darpa subterranean challenge: Technical overview and lessons learned, 2022.

[37] Marco Tranzatto, Takahiro Miki, Mihir Dharmadhikari, Lukas Bernreiter, Mihir Kulkarni, Frank Mascarich, Olov Andersson, Shehryar Khattak, Marco Hutter, Roland Siegwart, et al. Cerberus in the darpa subterranean challenge. Science Robotics, 7(66):eabp9742, 2022.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. CoRR, abs/1706.03762, 2017.

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems, 30, 2017.

[40] Sai Vemprala, Rogerio Bonatti, Arthur Bucker, and Ashish Kapoor. Chatgpt for robotics: Design principles and model abilities. Technical Report MSR-TR-2023-8, Microsoft, February 2023.

[41] Fanlong Zeng, Wensheng Gan, Yongheng Wang, Ning Liu, and Philip S. Yu. Large language models for robotics: A survey, 2023.