CogExplore: Contextual Exploration with Language Encoded Environment Representations

Harel Biggie*, Patrick Cooper*, Doncey Albin, Kristen Such, Christoffer Heckman
*Both Authors Contributed Equally
Department of Computer Science, University of Colorado Boulder
Email: {harel.biggie, patrick.cooper, doncey.albin, kristen.such, christoffer.heckman}@colorado.edu

***Abstract*—Integrating language models into robotic exploration frameworks improves performance in unmapped environments by providing the ability to reason over semantic groundings, contextual cues, and temporal states. The proposed method employs large language models (GPT-3.5 and Claude Haiku) to reason over these cues and express that reasoning in terms of natural language, which can be used to inform future states. We find that by leveraging natural language, semantics, and tracking temporal states, the proposed method greatly reduces exploration path distance and further exposes the need for environment-dependent heuristics. Moreover, the method is highly robust to a variety of environments and noisy vision detections, as shown with a 100% success rate in a series of comprehensive experiments across three different environments conducted in a custom simulation pipeline operating in Unreal Engine.**

I. Introduction

Exploring unmapped environments is paramount to modern robotic applications, such as search-and-rescue and assistive robotics. While state-of-the-art methods have made incredible progress in exploring in challenging search and rescue scenarios [5, 46, 1], they typically rely on hand-engineered heuristics based on the geometry of the environment, e.g. optimizing for volumetric gain. Such hand-engineered features fail to incorporate the rich semantics and contextual cues of an environment [4].

Given the problem of exploring a previously unknown environment, we develop a framework, Contextual Exploration (Cog Explore), that leverages foundation models, or large language models (LLMs) with natural-language-based representations of the environment to perform navigation using both geometric and contextually rich features, as well as temporal states.

To effectively leverage natural language, we must ground language utterances to the physical. Various approaches have been used to associate language with the physical domain, ranging from probabilistic graph-based structures [43, 20, 26] to end-to-end learning-based methods [14]. We represent the grounding problem as a probabilistic set of planning points, object points, and language priors obtained from a set of vision models. Our framework then selects the next best location to explore given this set of priors and a goal. Specifically, we leverage foundation models [7] [23] which have shown the ability to exhibit remarkable contextualization and reasoning by utilizing efficient next token predication. We validate CogExplore's exploration capabilities with 210 simulations of 45 minutes each operating in 3 different environments across 7 different tasks and show the method is capable of performing temporal, geometric, and semantic reasoning.

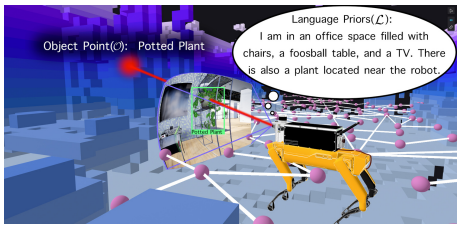

Fig. 1: Spot observing features of its environment for the prompt "Find a garden."

II. Related Works

A. Robotic Exploration

Graph-based approaches have proven effective in robotic exploration [43, 20, 26, 28, 36, 24, 27, 50]. These methods construct topological representations of the environment to guide exploration and planning. Recent works have incorporated semantics into these graph-based frameworks to enhance performance [11].

In addition to graph-based techniques, neural networks have been applied to robotic navigation, often serving as heuristics [31]. Leveraging advancements in computer vision, modern perception models have been integrated with traditional exploration approaches, leading to improved capabilities [17, 21].

B. LLM-Powered Robotic Navigation

LLMs have emerged as powerful tools for robotic navigation and manipulation. By encoding scene dynamics and task specifications in natural language, LLMs enable robots to reason about their environment and objectives at a high level [51, 2].

LLMs can act as agents capable of decomposing and executing complex tasks through modular prompting [30, 18, 22]. They have demonstrated proficiency in code generation for robotic applications [32, 47, 35, 38]. The combination of embodied agents for low-level skills and LLMs for high-level reasoning has shown promise [6].

End-to-end neural network approaches such as LATTE [8] and CLIPort [42] map natural language intentions to robot actions. HULC [34] combines language conditioning and

semantic knowledge for efficient imitation learning. While this approach works well in specific domains, it is not able to learn from the voluminous unsupervised datasets LLMs are trained from.

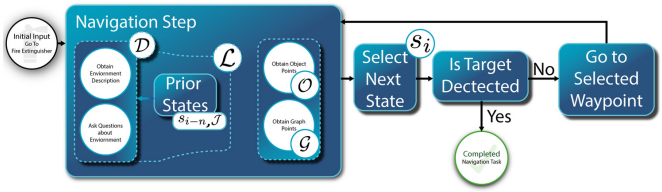

Fig. 2: CogExplore System Diagarm

Grounding natural language instructions to the physical environment is crucial for LLM-powered robotic navigation. Neuro-symbolic approaches [33] and learned traversability functions [39] have been explored for this purpose. PromptCraft [49] tackles the challenge of describing complex navigation tasks through prompt engineering. ORION [12] demonstrates personalized, interactive object navigation using natural language.

While LLMs excel at reasoning over unstructured data [16, 45, 44, 52] using transformer architectures [48], guiding them to desired outcomes and maintaining temporal coherence remains an open problem. Recent works integrating LLMs with object scene representation transformers for task and motion planning [6, 14, 38] still lack the interpretability and guarantees of traditional methods [36].

The semantic reasoning abilities of LLMs have been leveraged to guide exploration and planning in novel environments [40, 41]. LLMs have also been utilized for semantic grounding in complex outdoor environments using contextually relevant instructions [41]. Our method introduces new capabilities to this body of work, allowing LLMs to selectively encode relevant states based on environment descriptions and then reason directly over these natural language-encoded states, along with temporal states derived from a log of past movements.

III. Methodology: CogExplore

Our proposed exploration framework, CogExplore, represents relevant features of the environment as natural language strings. Formally, we represent possible states for the robot as, $\mathcal{S} = \{s_1, s_2, ..., s_n\}$ where s_i is a possible state for the robot. At each state, the robot has a set of planning graph points $\mathcal{G} = \{g_1, g_2, ..., g_k\}$, a set of object points $\mathcal{O} = \{o_1, o_2, ..., o_m\}$ and a set of language priors $\mathcal{L}$. Graph points represent traversable areas of the environment (sampled from [13]), and we represent them as a list in text from $\mathcal{G}_i = \{x, y, z\}$. Object points are represented similarly but with a corresponding label and probability ($\mathcal{O}_i = \{x, y, z, c, p\}$ where c is the class and p is the confidence represented as a probability), obtained from an open vocabulary object detector. Specifically, we utilize YoloWorld [10] and Segment Anything [25] to project open vocabulary detections into 3D points. Examples of $\mathcal{G}, \mathcal{L}$, and $\mathcal{O}$ can be seen in Figure 1 and full details of, the open vocabulary 3D detection system can be found in Section VIII-A of the Appendix. The language priors, $\mathcal{L}$, consist of three components; environmental descriptions, prior robot states, and the justification for choosing the next state. Environment descriptions $\mathcal{D}$ are obtained from a series of questions generated by foundation model and answered using a multimodal visual question and answering model (VQA) called LLaVA -1.5 [29], as shown in Figure 2. Prior states (s_{i-n}) and the justification for choosing the state ($\mathcal{J}_{i-n}$) and the full set of language priors is $\mathcal{L} = \{\mathcal{J}_{i-n}, s_{i-n}, \mathcal{D}\}$ along with the model's justification for selecting the state. In order to select the next best state, we can model the task as a maximum likelihood estimate:

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

At each iteration, the foundation model is asked to direct the robot to the state that is most likely to find the goal state, s_g given the robot's past states and any new observations. The process repeats until the robot arrives at s_g.

A. Exploration Planner

CogExplore's autonomous navigation uses a state-of-the-art graph-based exploration planner with frontier point finding [13] to help the language model select reliable states. Realtime mapping is performed using the a voxel-based map [19]. At each planning iteration, a set of sparse global graph points and dense local points are generated. We employ a 2D Gaussian function to sparsity the distribution of these points, where weights are assigned based on their proximity to the robot's current location. The result $\mathcal{G}$ is a distribution of low-density points in distant areas with a high concentration of points near the agent allowing CogExplore to plan with high granularity locally while still having the ability to explore regions of interest that are further away.

B. Foundation Model Prompting Schemes

We rely on language foundation models to perform four tasks: generating questions about the environment, generating object labels, compressing state information, and performing state selection. Foundation models are highly sensitive to the particular presentation of the underlying information [9]. To assist the model's geometric reasoning, we label some of the graph points ($\mathcal{G}$) as new if they are in an unexplored region. A point is labeled as new if it is beyond a certain distance threshold from all existing points in the graph.

For each iteration of the exploration cycle, the LLM is given a form to fill out, which creates a set of fields for the next waypoint output, as well as fields for a characterization of the environment and a justification field to insert reasoning for the selection of the particular waypoint it chose. Once the generation is complete, the LLM is queried again to compress the prior output into a concise state of 50-100 words ($\mathcal{J}$) these compressed states with justifications enable CogExplore to reason over its past states, facilitating a contiguous exploration

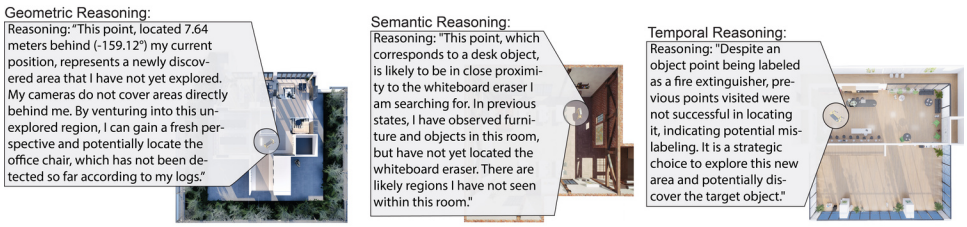

Fig. 3: Example Runs Demonstrating Varieties of Reasoning

process. Each of these logs is appended to a memory window with some fixed length, in practice we discovered a length of 10 was appropriate for both GPT-3.5-Turbo and Claude Haiku. Full prompts and information on prompt engineering can be found in the Appendix.

IV. Experimental Results

A. Experimental Setup

We evaluate the performance of CogExplore's ability to explore and find a target object using a simulated Boston Dynamics Spot in three different environments using Unreal Engine [15]. We compare CogExplore against a A^* direct path metric and the Vision-Enabled Frontier Exploration Planner (VEFEP) described below.

Two office scenes (Office 1, Office 2) and a more complex school environment (School) were designed, which provided the structure for a total of 7 object-retrieval scenarios to be tested, as shown in Table I. In Office 1, there were three tasks involving locating a fire extinguisher in different locations, denoted as FE1, FE2, and FE3. Office 2 featured two tasks: finding a coffee table (CT) and an office chair (OC). In School, the tasks were to locate a whiteboard (WB) and a bookshelf (BS).

For each task, we simulate the navigation capabilities of a Boston Dynamics Spot and allow the robot to explore for 45 minutes. A run terminates once the target object is found. We conduct 15 trials per task on Azure "NC64as_T4_v3" instances with 4 Nvidia T4 GPUs. We utilize 3 GPUs per simulation, one for the unreal simulator, one for the object detector, and one for the VQA model. We note that for Office 1, and the school scene, the simulator runs in real-time. While in Office 2 the simulator runs in half-time due to higher quality graphical assets.

We compare the performance of CogExplore running with Anthropic's *claude-3-haiku-20240307* (CE-H) and OpenAI'S *gpt-3.5-turbo-0125* (CE-3.5) as the backend models to a baseline Vision-Enabled Frontier Exploration Planner (VEFEP). VEFEP calculates the exploration gain heuristic g for a given path σ_i by summing the volumetric gain $\mathbf{VG}$ weighted by a function for each vertex in the path [3]. Parameters are based on the ones used by the authors in [13]. This planner explores until it detects the target object using the same 3D open Vocabulary detector used by CogExplore.

Quantitative Results: From Figure 4 we can see that in Office 1, the smallest environment, VEFEP, CE-H, and CE-3.5 all have similar mean path lengths. As the environment size increases in Office 2, making the exploration tasks more complex, we observe that on the office chair task both CE-H and CE-3.5 outperform VEFEP. We also note that VEFEP, timed out in 3 out of 15 of the runs as noted in Table I. VEFEP had the lowest median run length in the coffee table task. However, it also failed to find the table in 6 out of 15 experiments, as shown in Table I. In the whiteboard eraser task, both CE models significantly outperform VEFEP, which also had 5 out of 15 failures or a 33.3% failure rate. Of note, is that *none* of the CE methods failed at the exploration task in any of the environments.

Qualitative Results: In Figure 5 we observe that in the Whiteboard Eraser task CE-H and CE-3.5 only enter the room with the eraser once whereas VEFEP enters, exits and proceeds to loop around the environment. Similarly, in the Go to the Office Chair task CE-H and CE-3.5, immediately go to the chair and take more efficient paths to get there. In contrast, VEFEP heads outside and explores the areas significantly before returning to indoors to find the chair.

V. Discussion

Our results highlight the robustness of CogExplore's exploration abilities with a *100% success rate* across all environments with both GPT and Haiku variants, in contrast to VEFEP's 15.2% overall failure and 40% failure rate at the coffee table task. Moreover, CogExplore's use of natural language justifications for each state selection allows us to directly probe why the framework is acting in a certain manner and understand the rationale behind the more efficient exploration paths. From these justifications, we can see that the model is capable of performing geometric reasoning, semantic reasoning, temporal reasoning, and fault-tolerant reasoning.

Geometric Reasoning. We see geometric reasoning on the right side of Figure 3 where the model explicitly chooses a point behind the robot. The robot camera positions are encoded into the prompt and the model is aware that the robot can not detect objects, directly behind it, since there is no rear camera. This variety of intuitive geometric reasoning directly contributes to the shorter exploration path lengths demonstrated by CogExplore.

Semantic Reasoning. Another key factor that contributes to robust and efficient performance is the model's ability to semantically reason over cues in the environment. In the center of Figure 3 we see the model choosing a nearby point because a desk was found that could contain a whiteboard eraser

Office 1 (572 m^2): Figure 11a			Office 2 (1450 m^2): Figure 11b			School (1287 m^2): Figure 11c		
Task	**Direct Path (m)**	**VEFEP # Timeout**	**Task**	**Direct Path (m)**	**VEFEP # Timeout**	**Task**	**Direct Path (m)**	**VEFEP # Timeout**
Fire Extinguisher 1 (FE1)	25.2	0	Office Chair (OC)	33.7	3	Whiteboard Eraser (WE)	26.1	5
Fire Extinguisher 2 (FE2)	27.1	0	Coffee Table (CT)	26.2	6	Book-shelf (BS)	13	2
Fire Extinguisher 3 (FE3)	15.6	0						

TABLE I: Simulation Environments and Corresponding Tasks. Each run was given 45 minutes of simulation time to complete. None of the CogExplore runs timed out, and the VEFEP timeouts are shown here.

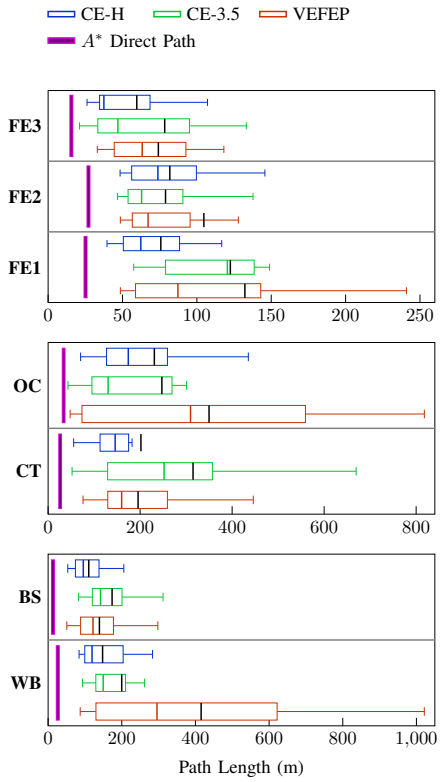

Fig. 4: Path length comparisons for each method (CE-3.5, CE-H, VEFEP) on completing each of the seven tasks. The A^* Direct Path from starting position to object position is also shown. Medians are shown in the box color and means are shown in black.

(the desired object in the exploration task). Both objects are commonly located in classroom and office settings, hence they share semantic similarities. Traditional frontier-based planners continue exploring other areas based on the original heuristic for volumetric gain despite these cues, which is also evident by the path taken by VEFEP in Figure 5.

Temporal Reasoning. Beyond reasoning over the current state, efficient path exploration requires an agent to reason over its past states. In CogExplore's case, we note the robot venturing to a new area despite having a point labeled as a fire extinguisher (the target exploration object). The robot already explored the nearby area in detail in a prior state, and the target object was never reached. The model correctly concludes the detection was in error. CogExplore justifies this change based on an understanding that object detections can be fallacious, which also highlights the framework's ability to perform fault-tolerant reasoning and remain robust to noisy sensor observations. In contrast, it was observed that the VEFEP planner would frequently stick around objects as the location of the 3D object projections were refined (as new observations were obtained), despite this not being an effective strategy after a few iterations.

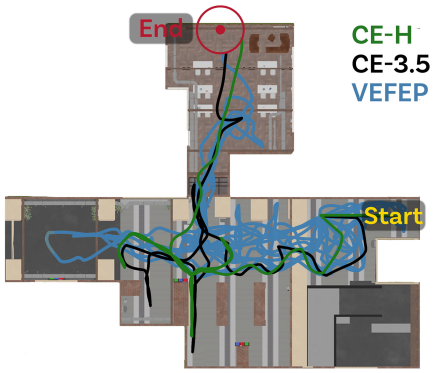

Fig. 5: Note that the VEFEP revisits the same places several times before eventually locating the goal object, whereas the CogExplore is able to more effectively reason over unexplored regions.

VI. Conclusion

CogExplore is a comprehensive framework for leveraging foundation models in robotic exploration tasks. The framework offers the ability to perform exploration that is geometrically, semantically, and temporally aware while remaining resilient to failures in grounding. Performance is evaluated across 210 photorealistic simulations in 3 different environments with 7 different exploration tasks. Our findings reveal that as the complexity of the exploration task increases, in terms of environment size and trajectory length, CogExplore's performance advantage over a vision-enabled exploration planner becomes more pronounced. This positive correlation shows that CogExplore is particularly well-suited for handling challenging navigation scenarios.

References

[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, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[3] Abraham Bachrach. Trajectory bundle estimation for perception-driven planning. 2013. URL https://api.semanticscholar.org/CorpusID:36535521.

[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. URL https://proceedings.mlr.press/v229/biggie23a/biggie23a.pdf.

[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. URL https://arxiv.org/abs/2005.14165.

[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] Jaime Crespo, José Carlos Castillo, Oscar Martinez Mozos, and Ramon Barber. Semantic information for robot navigation: A survey. *Applied Sciences*, 10(2):497, 2020.

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

[13] 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. doi: 10.1002/rob.21993.

[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] Epic Games. Unreal engine. URL https://www.unrealengine.com.

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

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

[18] Tanmay Gupta and Aniruddha Kembhavi. Visual programming: Compositional visual reasoning without training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14953–14962, 2023.

[19] Armin Hornung, Kai M Wurm, Maren Bennewitz, Cyrill Stachniss, and Wolfram Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34:189–206, 2013.

[20] 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.

[21] Chenguang Huang, Oier Mees, Andy Zeng, and Wolfram Burgard. Visual language maps for robot navigation. *arXiv preprint arXiv:2210.05714*, 2022.

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

[23] 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.

[24] Sertac Karaman, Matthew R Walter, Alejandro Perez, Emilio Frazzoli, and Seth Teller. Anytime motion planning using the rrt. In *2011 IEEE international conference on robotics and automation*, pages 1478–1483. IEEE, 2011.

[25] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[26] 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.

[27] Xiaodong Lan and Stefano Di Cairano. Continuous curvature path planning for semi-autonomous vehicle maneuvers using rrt. In *2015 European control conference (ECC)*, pages 2360–2365. IEEE, 2015.

[28] Steven M LaValle et al. Rapidly-exploring random trees: A new tool for path planning. 1998.

[29] 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. URL https://llava-vl.github.io/blog/2024-01-30-llava-next/.

[30] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[31] Thi Thoa Mac, Cosmin Copot, Duc Trung Tran, and Robin De Keyser. Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems*, 86:13–28, 2016.

[32] Aman Madaan, Shuyan Zhou, Uri Alon, Yiming Yang, and Graham Neubig. Language models of code are few-shot commonsense learners. *arXiv preprint arXiv:2210.07128*, 2022.

[33] Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B Tenenbaum, and Jiajun Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019.

[34] 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.

[35] Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2022.

[36] 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.

[37] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[38] 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.

[39] Dhruv Shah, Benjamin Eysenbach, Gregory Kahn, Nicholas Rhinehart, and Sergey Levine. Ving: Learning open-world navigation with visual goals. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13215–13222. IEEE, 2021.

[40] Dhruv Shah, Michael Robert Equi, Błażej Osiński, Fei Xia, Brian Ichter, and Sergey Levine. Navigation with large language models: Semantic guesswork as a heuristic for planning. In *Conference on Robot Learning*, pages 2683–2699. PMLR, 2023.

[41] 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.

[42] 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.

[43] 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.

[44] Stefanie Tellex, Nakul Gopalan, Hadas Kress-Gazit, and Cynthia Matuszek. Robots that use language. In *Annual Review of Control, Robotics, and Autonomous Systems*, volume 3, pages 25–55. Annual Reviews, 2020.

[45] 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.

[46] 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.

[47] Priyan Vaithilingam, Tianyi Zhang, and Elena L Glassman. Expectation vs. experience: Evaluating the usability of code generation tools powered by large language models. In *Chi conference on human factors in computing systems extended abstracts*, pages 1–7, 2022.

[48] 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.

[49] 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. URL https://www.microsoft.com/en-us/research/publication/chatgpt-for-robotics-design-principles-and-model-abilities/.

[50] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*, pages 146–151. IEEE, 1997.

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

[52] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

VII. Appendix

A. Overhead Path Views

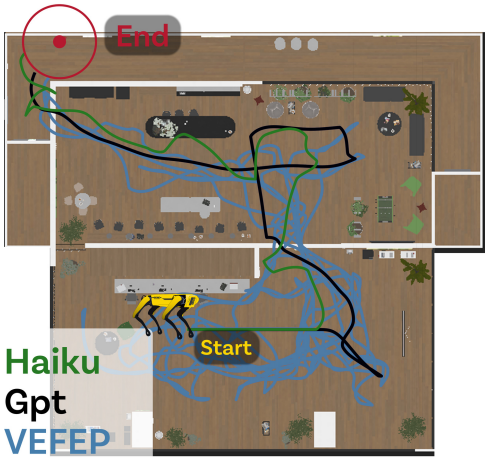

Fig. 6: Birds-eye view of Office Environment 1 (OE1) for task Fire Extinguisher 1 (FE1).

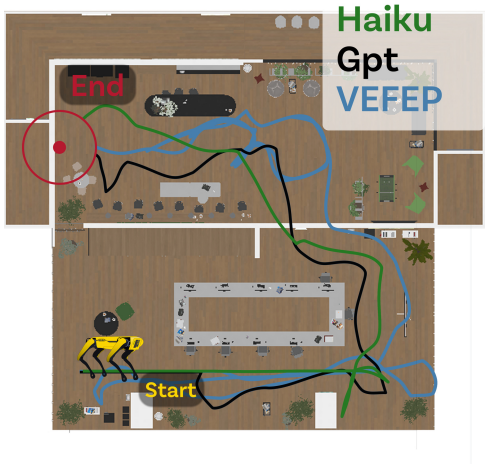

Fig. 7: Birds-eye view of Office Environment 1 (OE1) for task Fire Extinguisher 2 (FE2).

VIII. Unreal Renderings

Example renderings from each scene are shown in Figure 11

A. Open Vocabulary Object Detection Pipeline

Our open vocabulary detection pipeline is shown in Figure 12

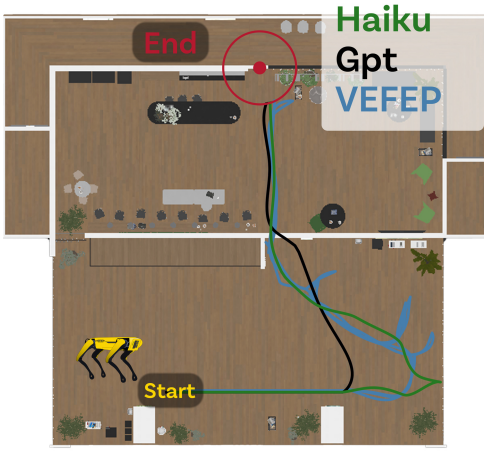

Fig. 8: Birds-eye view of Office Environment 1 (OE1) for task Fire Extinguisher 1 (FE3).

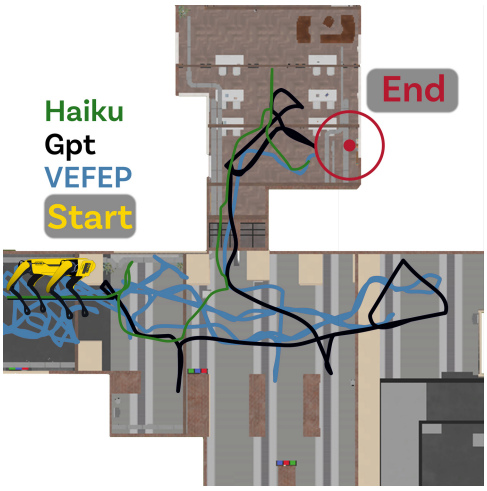

Fig. 9: Birds-eye view of School for task Bookshelf

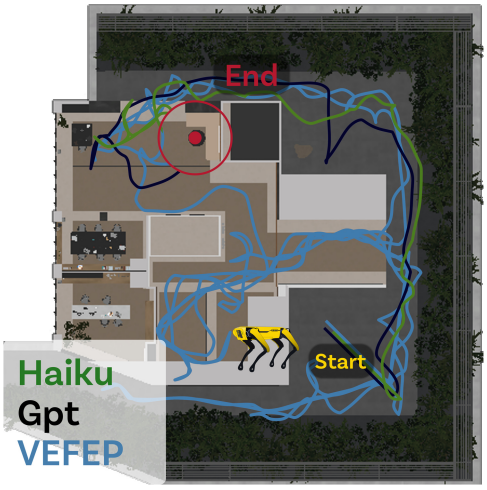

Fig. 10: Birds-eye view of Office 2 for task Coffee Table

(a) Office 1 (b) Office 2 (c) School

Fig. 11: Renderings from Unreal Engine Environments

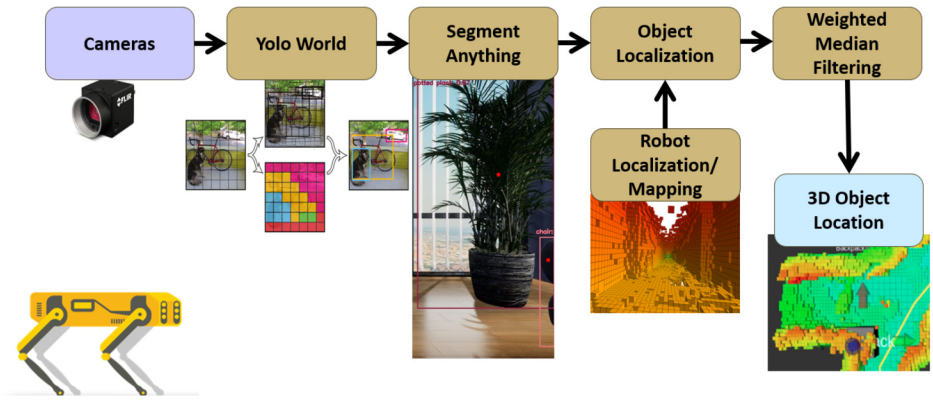

Fig. 12: Open Vocabulary Detection Pipeline. The pipeline uses YOLO World [10], a variant of YOLO [37] to perform 2D detections which are then segmented by Segment Anything [25]. We take the centroid of the segmented object and perform a ray cast to project the object onto the robot's Octomap [19]. Projections are filtered using a weighted median filter.

B. Prompts For Foundation Models

The full foundation model prompts for the foundation model are provided below. To guide both models toward more effective exploration, we added model-specific instructions at the end of the general prompt. These additions aimed to address model-specific issues where certain aspects of the general prompt were either being followed too closely or ignored.

In GPT-3.5's case, the model would occasionally select waypoints near windows when a potential object was visible through the transparent glass. To mitigate this, we explicitly reminded this model that windows might obscure routes to prospective targets, encouraging more practical waypoint choices. Claude Haiku was quite conservative in its exploration behavior especially before semantic cues were discovered. Explicit language was added to prioritize the exploration of new areas.

```
Your task is to create a list of 3 to 5 questions for a VQA (visual question answering model) that generate a clear
    and concise description of the enviornment that can later used by the robotic agent to navigate to the object
    or area described by this command: *INSERT_QUERY_HERE*
The description you elicit by providing questions to the VQA model is going to be passed to an operator who will use
    its content to make guesses about what objects in the environment to try to detect to assist with navigation.

VQA models are great at providing specific feedback to specific questions.

The robot you are guiding is the spot robot from Boston Dynamics.

The object or area we are after may be in plain view or may require exploration to discover. Do not presume the
    existence of any objects that you think may be in the scene.

For example, DO NOT ask any questions about the target area or object described in the command above. For example it
    is correct to ask if the target object is visible in the scene, but it is incorrect to ask for details about
    the target object since it may or may not be in the image.

Ask yes or no questions about whether an object exists or ask general questions about the kind of environment
    depicted within the image. DO NOT ASK ANY OTHER KIND OF QUESTION.

Your work will be added after two starter questions (do not include them in your response):
["What is the general setting of the environment: domestic, industrial, natural?", "What are the most prominent
    features visible in this environment?"]

Expand on these questions with 3 to 5 of your own. Do not replicate the work done by the above generic questions.
    Write your response formatted as a python list. That is a list denoted by brackets with each question separated
    by a comma. Only write this list. Do not include anything else in your response.

For example, a list with the two above questions included would look like: ["What is the general setting of the
    environment: domestic, industrial, natural?", "What are the most prominent features visible in this
    environment?"]

Note, your response must be a perfectly formatted list. You cannot include any other comments or characters in your
    response that are not precisely formatted in the same manner as the list example above. Make sure to include
    commas in between entries within the list. Make sure to include quotes around each entry within the list. These
    are strings within python, they require quotes.

Do not include any of the starter questions in your response. Just include the questions you have thought up.

The VQA model does not know anything about the position of the robot or that it is examining images from a robot.
    The questions must be straightforward and directly about the image itself.
```

Fig. 13: VQA Prompting

```
*INSERT_SCENE_DESCRIPTION*

Given you have this description of an environment we want to identify a set of objects that help us navigate a
    robotic agent to fulfill the goal: *INSERT_QUERY_HERE*

Given this information, generate a comprehensive list of all objects including the objects referenced in the prompt
    that may help the robotic agent fulfill the navigation request. This list will be used to guide an object
    detection machine learning model. So words in this list should be general and likely to be spotted within
the environment as we have characterized it in the scene description details above. You should look for specific
    individual items, such as tables, and plants that can be easily distinguished from the background. DO NOT look
    for general entities that could take up the entire frame such as wall, floor, path, road, or park. Lables like
    this should not be in your list.
Do not look for very large objects.

Always include the object or area you are searching for in this list. Rember this object detector is running in
    realtime on a moving robot.

This list should be between 1 and 5 objects in length. It should be formatted as a python list. For example, a list
    of the following objects would be precisely formatted as: ["chair", "table", "helmet", "tree"]
Note the first entry must be the goal object!

This list is the only thing that should be included in your response. The python list format must be followed
    precisely. Be sure to include quotes around the elements of the list.

Do not return anything besides this list.
```

Fig. 14: Object Detection Label Prompting

```
You are guiding a robotic agent to solve the following robotic navigation task *INSERT_QUERY_HERE*.
You are assisted by an exploration planner that produces potential new areas to explore in the form of graph and
    frontier points. Frontier points tend to be in areas you have explored less or have just discovered.
There is also an onboard object detector that produces 3D projections of points in space that correspond to objects
    from a given object list. Remember object points represent the centroid for an object and your planner will get
    you as close as it can to them.
Object points are often not perfectly labeled. Frontier points are calculated based on exploration potential. Object
    points will come from areas you have already observed.
Your objective is to select a point from among these within the scene, and determine what will serve as the robot's
    next waypoint.
You will only have waypoints in areas your robot can see with its lidar and cameras. So, it is important to probe
    areas that potentially offer vantages for views you have not yet seen. When you see new areas, you should
    explore them.
If you have been near a target for more than one call to this point selection logic, then it is not the correct
    target. The computer vision your robot has is not perfect but is tuned towards producing more false postives.
    Be adventurous in your exploration, travel to areas far way from you.
Remember your robot is continuously detecting objects and has a left, right and front cameras. When you select a
    point the planner will plan a direct route for your robot to take. If the target object is deteceted along your
    path, the robot will go to it and add an entry to the summary below.
Use the available information from the robot below along with any contextual information you have available to
    select the next point to navigate to.
The time since start of sim is:
*INSERT_CURRENT_TIME* seconds
The spot robot is now at:
*INSERT_CURRENT_POSITION*
Here is a description of the area spot is currently in:
*INSERT_SCENE_DESCRIPTION*
The robot has been instructed to search for these objects:
*INSERT_OBJECT_LIST*
You have the following list of points you may choose from (graph points are not labeled, frontier points and object
    points are). If a graph point is labeled as "new" then it is from a region you have discovered it in the last
    step and are now able to plan to.
You should go to new points! They represent unknown areas and new views for your cameras and lidar. You may select
    either a graph point or an object point.
*INSERT_FRONTIER_OBJECT_POINT_NUMBERED_LIST*
You have been asked to help in the past *INSERT_TOTAL_CALLS* times to inform this navigation task. You should never
    go to the same point multiple times. The simulation ends when you reach the target.
Here is a summary of the actions you performed in the last *INSERT_MEMORY_LENGTH* calls:
*INSERT_PRIOR_STATES*
*INSERT_INTERRUPT_DESCRIPTION*
Now that you know where you've been, try to explore areas you have not yet been to. Use the description of the
    environment in past states to go where the object is most likely. If you have no prior states, be aggressive in
    your exploration.
If you have been by a point for more than one call, you should move, even if it is the target object. The robot's
    vision makes mistakes. Do not go to the same point more than once! Check the coordinates (x,y,z) of the point,
    if you've gone near there before in any of these prior states, don't go there again.
Change points every time, even if the point is an object point which appears to be the target. Cycling back and
    forth between two areas will not give new observations. The environment is static if you have already been to
    an area your cameras should have found the object.
As you explore the scene, new points will become available for you to plan to. Try to select points on the edge of
    those available to you. Consider areas that will allow you to discover new points. If a point is far from the
    other points in your list and in your prior states, it is a good candidate.
Reason about the area of space defined by your points. Go to areas and rooms outside of spaces you have explored.
Avoid considering points near your current location if you do not believe they are near the target.
Do not worry about collisions. All points are safe to travel to. Explore the space aggressively. Seek out the object
    and trust the robot's vision to find it.
If you have a point or points which are labeled as a new point, you should go to one of these points, your task is
    to to explore with contextual awareness. These new points are newly discovered and represent a high potential
    for discovering your target.
Recall, you should consider x, y, and z when deciding where to go for both the object locations and the potential
    frontier points you want to go to. Do not just look at one coordinate! Consider both x and y equally! Euclidian
    distance should be used, we want to be close to the target object across all dimensions.
The distance field in your point list is the Euclidian distance from the robot to the object.
NEVER GO TO THE SAME AREA TWICE IN A ROW!!!
*INSERT_SPECIFIC_MODEL_INSTRUCTIONS*
Based on the information above fill out the form below (if you select an object, make sure to explain why that
    object is related to the navigation task. Reference past states in your reasoning. You should always navigate
    in reference to where you've been. Reason about the space you are in. Reference the objects you see and
    categorize your environment.):
I need to select a waypoint from a numbered list of graph points, frontier points and object points. The point I
    have selected is point number: [insert the graph, frontier or object point numeral number here]. I am selecting
    this point because I believe it makes strategic sense to get me closer to solving my navigation task
    "*INSERT_QUERY_HERE*". My environment can be described as [describe the environment]. My reasoning is that this
    point [insert your reasoning here].
```

Fig. 15: Explore Iteration Prompting

```
If you see the object go directly to it.

Avoid going to regions that you have been to before. Consider how close the point you are selecting is with those
     listed in your prior states.
If it is nearby, don't return to it.

Use your vision to go to areas that are rich with objects like the one you are after.

Newly discovered areas that are far away are highly advantageous
if they are far from where your robot has traveled before.

Use the points to gain a sense of space. Travel to regions where you have not yet been.

Don't be tricked by windows, if you can't get to interesting objects, they might be behind a window.

GO TO AREAS THAT ARE FAR AWAY! DON'T STAY IN ONE REGION!!
```

Fig. 16: GPT-3.5 Turbo Specific Prompt Additions

```
Avoid looping between areas you have already been to. Try to find new areas so you will generate graph points for
     them.
If there is an area you have not been to yet, go to it. Never double back, keep exploring nearby unknown areas.

Consider each prior state, do not return to anywhere near these locations.

Newly discovered areas that are far away are highly advantageous if they are far from where your robot has traveled
     before.

Use the points to gain a sense of space. Travel to regions where you have not yet been.

Always prioritize new points when they are available.

GO TO AREAS THAT ARE FAR AWAY! DON'T STAY IN ONE REGION!!
```

Fig. 17: Claude Haiku Specific Prompt Additions