Tell Me Where to Go: A Composable Framework for Context-Aware Embodied Robot Navigation

Anonymous Author(s)
Affiliation
Address
email

Abstract: Humans have the remarkable ability to navigate through unfamiliar environments by solely relying on our prior knowledge and descriptions of the environment. For robots to perform the same type of navigation, they need to be able to associate natural language descriptions with their associated physical environment with a limited amount of prior knowledge. Recently, Large Language Models (LLMs) have been able to reason over billions of parameters and utilize them in multi-modal chat-based natural language responses. However, LLMs lack real-world awareness and their outputs are not always predictable. In this work, we develop a low-bandwidth framework that solves this lack of real-world generalization by creating an intermediate layer between an LLM and a robot navigation framework in the form of Python code. Our intermediate shoehorns the vast prior knowledge inherent in an LLM model into a series of input and output API instructions that a mobile robot can understand. We evaluate our method across four different environments and command classes on a mobile robot and highlight our framework's ability to interpret contextual commands.

Keywords: Natural language, navigation, contextual navigation

1 Introduction

Humans have the remarkable ability to navigate through unfamiliar environments, e.g., in a town or through a building, by relying solely on our priors and descriptions of the environment [1]. Motivated by the difficulties in directing robots in collaborative teams with humans such as those used in search and rescue [2, 3, 4, 5] operations, we aim to develop a framework that allows humans to provide high-bandwidth instructions to robots in the form of natural language.

To achieve this, it is generally required that a robot would associate natural language utterances to the physical world using sensing modalities onboard the robot, a process known as grounding. However, unlike humans, robots are not currently capable of integrating prior experiences into a vast wealth of priors to aid this association.

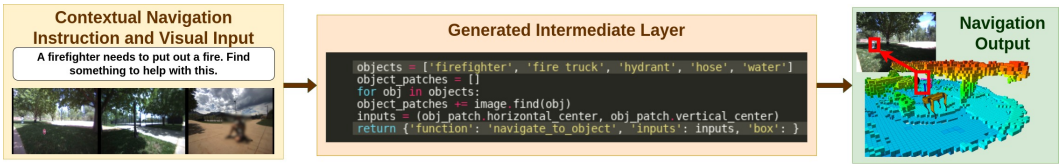


Figure 1: Contextual navigation example for a firefighting situation. We see our compositional framework generates code for the robot to both find and then plan to a fire extinguisher.

Recently, Large Language Models (LLMs) [6] have been able to reason over billions of parameters and utilize the results in domains such as dialogue-based responses [7, 8, 9], code generation

[10, 11, 12], and multimodal visual question and answering (VQA) tasks [13]. This type of unsupervised learning shows incredible potential for the generalization of prior knowledge but these models lack real-world experience. A lack of physical experience is perhaps one reason why transitioning techniques developed through research in VQA and view-based navigation into embodied agents has not kept pace with the research in these fields.

Additionally, with the current generation of LLMs, it's inherently challenging to interpret the rationale behind the outputs that are generated by the model. In this work, we address both the grounding and transparency issues using a composable framework designed for robot navigation. Our pipeline extends the reasoning framework for visual inference presented in [14] to an embodied robotic agent. With our composable framework, we leverage LLM priors (GPT-3.5), state-of-the-art object detectors [15], and classical robotic planning algorithms [16, 17, 18] to perform zero-shot natural language based navigation in four unique environments.

Specifically, our contributions are as follows:

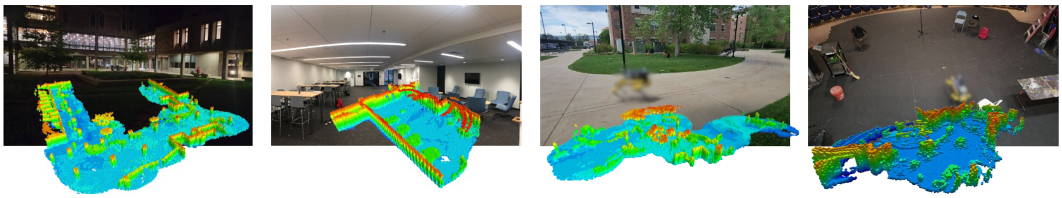
- We extend the concept of modular neural networks and define a new framework for composable robot navigation. Our framework requires a minimal uplink for the robot since all of our planning, mapping, and localization is performed onboard.
- We evaluate different 2D input representations to determine an effective way to extract spatial and conceptual knowledge from LLMs.
- We perform extensive real-world experiments in a variety of environments and show that our framework is able to navigate to landmarks based on natural language. Furthermore, the framework is able to deduce appropriate navigational goals from the context of a sentence.

2 Related Works

Grounding Language: Associating natural language with an embodied agent requires grounding utterances to the physical world in which the robot operates. Various approaches have been used to associate language with the physical domain ranging from probabilistic graph-based structures [19, 20, 21] to end-to-end learning-based methods [22]. Graph-based approaches have shown promising results but their generalizability is limited to a fixed training corpus. On the other hand, LLMs have proven to be adept at reasoning over large amounts of unsupervised training data [23, 24] using transformer-based backends [25]. These models also known as foundational models due to their comprehensive knowledge have recently been leveraged for task and motion planning using reinforcement learning on a set of skills [26], and in an end-to-end fashion [22] 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 share [28].

Modular Neural Network Frameworks and Code Synthesis: Modular Neural Networks [29, 30, 31] are remarkably adept at answering questions about images in a task commonly known as VQA [32]. Until recently many of these methods were limited in their ability to generalize to other domains due to the difficulty of generating interfaces between modules. In [14] this was addressed by using recent code generation techniques [33] from LLMs to create these interface layers. Specifically, these modular frameworks enable the generation of code which dictates the interaction between robust object detectors [15, 34, 35, 36, 37], depth estimators [38, 39, 40]. What's more remarkable is that these LLMs can also utilize existing functions inside of the Python language such as sorting and conditionals without any additional training [41, 42, 12, 24]. Based on these exceptional findings we aim to extend the use of these modular concept-learning style frameworks to the robotic navigation domain.

Path Planning for Robotic Navigation: Path planning for robotic navigation and exploration is a long-studied research area [43, 44] with solutions ranging from sampling-based methods[16, 45] to graph-based lattice structures [46]. Variations of these probabilistic approaches are being used for self-driving cars [47], exploration [18, 4, 3, 5] in complex environments. These approaches



(a) Courtyard (b) Lobby (c) Outdoor (d) Theater

Figure 2: Full maps and representative images from the four testing environments.

benefit from probabilistic guarantees on convergence and optimality [16] and adaptability to new environments making them natural choices to utilize in language grounding framework.

Embodied Language-Based Navigation: Recently a few other works have leveraged foundational models for embodied navigation. Clip on Wheels (CoW) is proposed in [48] which combines text and image caption models [49] with frontier-based exploration algorithms [50] to perform object exploration in simulation. In [51] landmark-based navigation is performed using pre-trained visual [48], language, and navigation modules [52]. Visual Language Navigation Maps (VLMaps) [53] learn a spatial language representation by combining RGB video feeds, code generation, and foundational models. SayCan [54], and Palm-E [22] utilize the PALM [55] language model to perform skill-based navigation and end-to-end navigation respectively. In general these methods are not tested on contextual examples, and some require the preprocessing of maps; in contrast, we test on situated examples with an experimental platform and require no situated priors.

3 Methodology

We develop a framework that leverages the rich contextual priors of LLMs and creates an executable code layer that interfaces with planning algorithms running on an embodied agent. Drawing on paradigms from modular neural networks [56, 57, 14], and concept learning techniques [58], we define a modular system that takes inputs at various stages in order to fully define a final navigational output for our system. This enables the use of arbitrary intermediate and substitutable layers which can also be executed on a distributed computing infrastructure. Formally, we have a collection of inputs ψ composed of a visual input $v \in V$ an RGB image or multi-perspective collection of images, $m \in M$ a volumetric map of the space to be navigated, and $c \in C$ a natural language navigation command. Our system takes initially as input a command c to generate code γ_c. This code consumes v in order to resolve the grounding problem, i.e., to determine which object is being referred to in c for navigational instructions, as well as m in order to emplace the object into the world around the robot. This outputs $i = \theta(v, m|\gamma_c) \in \mathbb{R}^3$, a 3D waypoint to which navigation will commence in order to output $\mathcal{P}$ a robot trajectory which is a continuous function in $\mathbb{R}^3$. A graphical overview of the framework can be found in Figure 3.

Input Representations: Our visual input v is in the form of an RGB image, either a semi-panoramic view or three separately labeled spatial images (left, front, right). Typical inference methods over RGB images [36, 35, 15] reason in 2D over the image. For embodied navigation, we need to reason in 3D and associate different camera viewpoints with their associated 3D spatial relations. Embodied navigation introduces spatial relations that are difficult to reason over in 2D such as "behind", "in front of", or "on your right". To determine an appropriate input representation we evaluate the difference between sending in a concatenated image of all viewpoints (semi-panoramic) from the agent or sending in each frame separately with a spatial definition, i.e., right, front, or left. We find that spatial reasoning is best performed on a concatenated image and we present the details in Section 4.

Intermediate Layer: We generate θ using recent advances in code generation models [10, 42]. We provide a functional set of navigation instructions in the form of a Python API for the code generation model. The full prompt containing the API can be found in the Appendix. These instructions

include specifications for how to perform visual inference on the input v using a similar paradigm found in [14]. These API instructions include directions for finding an object, checking if an object has a certain property, etc. Additionally, we provide an API specification for interfacing with our geometric planner, resulting in code generated based on the natural language prompt γ_c.

Planning: We use a graph-based planner first presented in [18, 4] where we provide a navigational endpoint. Specifically, we create a function that takes in the center coordinate p of an object in image space and projects it as a landmark l on a 3D map. The planner uses a sample-and-project approach similar to the one described in [59, 18] for terrain avoidance. Using this strategy, a graph of plans $\mathcal{G}$ is constructed by sampling points parameterized by the robot's width and length [16]. When a 3D waypoint is input to the planner we select the best path $\mathcal{P}$ from $\mathcal{G}$, for the robot to follow. If no path exists, we plan to the boundary of the graph and resample until the goal is reached.

Waypoints are passed to the planner using the output i from the intermediate layer. Specifically, image coordinates are translated into waypoints by executing θ and associating the result onto a 3D map m [60] using ray casting. The map m is generated online using [61] and the translation layer θ creates the necessary code to translate between the inputs c, v, m as and the output i. We then plan to the waypoint by selecting the best path from $\mathcal{P}$.

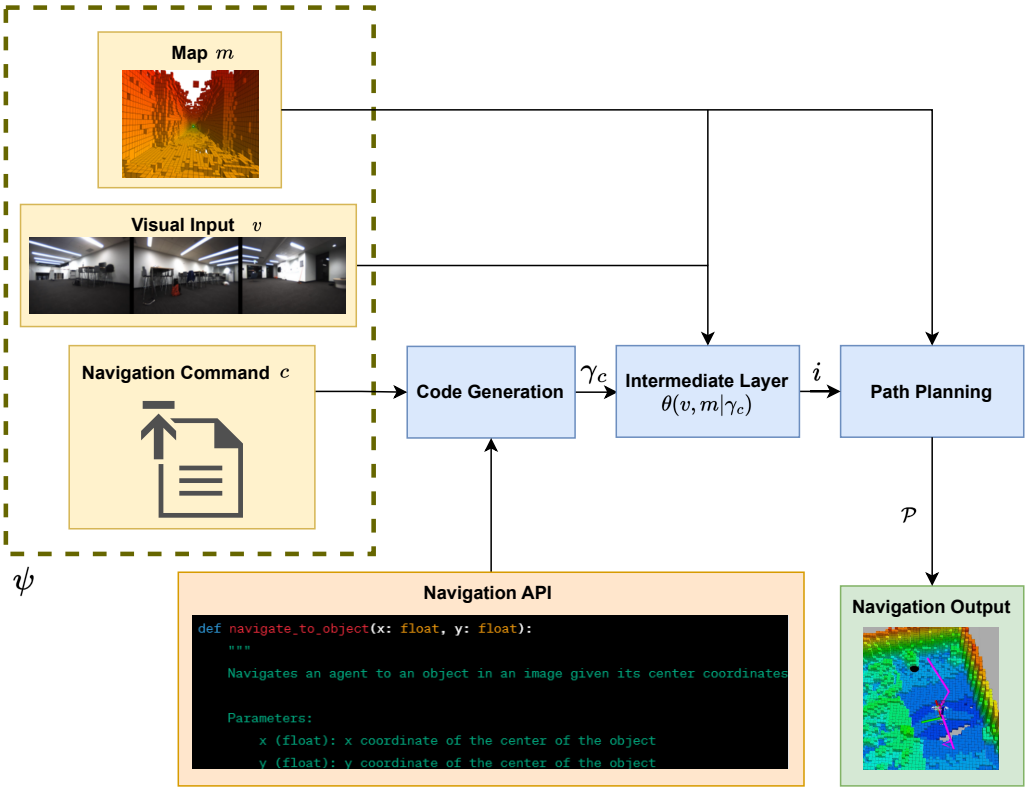


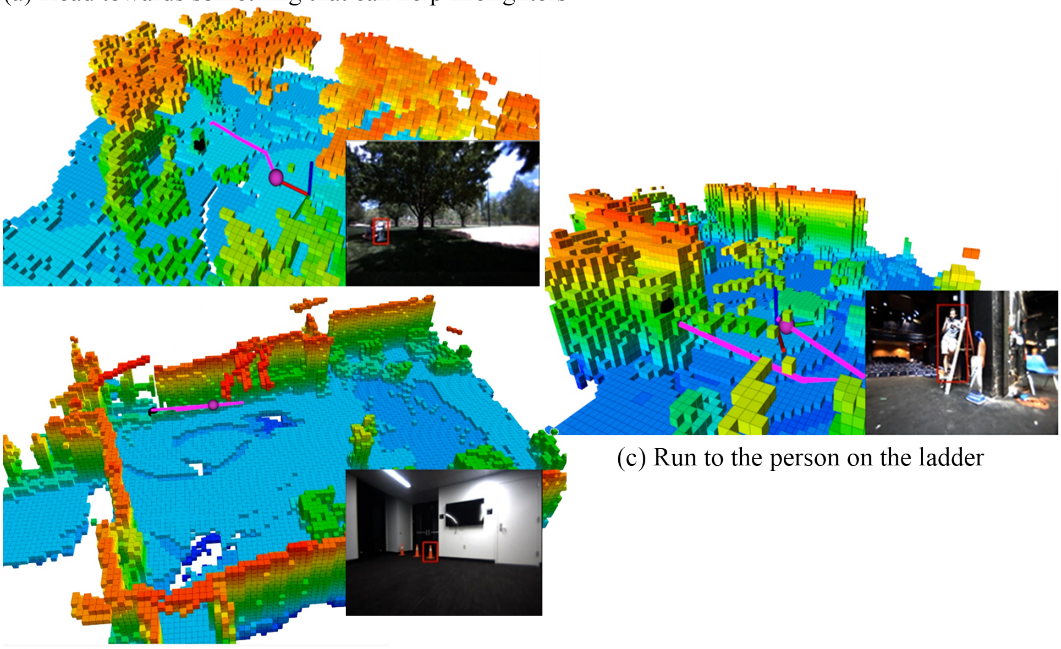
Figure 3: A graphical overview of the full system framework.

4 Experimental Results

We run two sets of experiments on a Boston Dynamics Spot equipped with a custom sensing suite consisting of a 3D 64 beam Ouster lidar, an imu, and 3 RGB cameras providing a semi-panoramic view of the environment. Our first experiment is designed to determine the best input representation for the visual layer and the second set tests the ability of our system to perform navigation in a wide variety of real-world environments.

For all of our experiments, we leverage ideas from human concept learning as in [58] to categorize our sentences into four categories: Generic, Specific, Relational, and Contextual. Generic sentences are simply sentences that imply "Go to something", e.g. "Walk to the backpack.". Specific sentences include a distinguishing piece of language such as a color attribute that directs the robot to one specific object in the scene. For example, if there are two backpacks, one red, and one black, then a specific sentence would be "Drive to the black backpack." Relational sentences are any sentences that describe spatial relationships between objects in the scene such as, to the right of, on top of, etc. An example in this category is "Move to the backpack on top of the chair." Contextual examples, require the robot to interpret the navigational goal based on background information. For instance, the sentence "Find something that can carry water." requires the robot to know that a cup or a bucket can hold water. Our full list of sentences can be found in the Appendix.

(a) Head towards something that can help firefighters



(c) Run to the person on the ladder

(b) Walk to the right most cone

Figure 4: Example sentences, visual detections, maps, and planned paths.

Encoding spatial relationships into a model which has no concept of the physical world presents significant challenges. We determine an effective scheme for inputting three different viewpoints into the framework through empirical evaluations. Take the sentence, "go to the chair on the right" implies only the right half of the total FOV should be investigated but "Go to the backpack that is to the right of the chair." could be in any image. To solve these limitations we use two different sets of input representations, the first is all three images stitched together with padding in between frames (**A**). In this case, we instruct the LLM on the order of the images (left, front, right) and let the model handle the spatial reasoning. In the second case, we process the frames individually (**B**) and let the model decide which frames to look at. We explicitly prompt the model with example code snippets that solve the difference between "to your right" and "right of" styles of language. at configuration. In this set of experiments, we evaluate the correctness of the generated code as well as if the code produced the correct object in the map m. This scene was created in a classroom setting with good lighting conditions as the emphasis was to evaluate the effect of the input representation. Results for successful code generation and object detections for the two input schemas are summarized in Table 1.

From Table 1 we find that concatenating images significantly outperforms sending in the three frames separately. As expected, in the generic navigation case we are able to achieve 100% suc-

Category	Count	**A** (%)	**B** (%)
Generic	12	100	100
Specific	12	91.67	66.67
Relational	15	86.67	53.33
Contextual	11	81.82	45.45
Total	50	**90**	66

Table 1: Results showing the difference between the concatenated input representation **A** and the sequential representation **B**.

cess in generating the intermediate layer and for object identification. We see that the individual frame configuration (**B**) begins to break down when reasoning over specific objects. Specifically, the model fails to reason over spatial relations and object ordering when objects appear in more than one frame. For example, taking the sentence "Go to the middle outlet" only works when the three outlets are present in the same camera frame. If the middle outlet is in the front frame but the right outlet is in the right camera frame, this method fails. We explicitly see this in the generated code:

```
outlet_patches.sort(key=lambda x: x.horizontal_center)
middle_outlet = outlet_patches[len(outlet_patches) // 2]
```

where each detected outlet is ordered based on its horizontal coordinate in the image. Since all outlets are merged into the same list the images have overlapping coordinates causing these failures. Configuration **B** struggles with relational sentences between objects for similar reasons.

In our second set of experiments, we evaluate the ability of our framework to navigate in real-world environments. The four environments shown in Figure 2 range from an indoor lobby setting to a dark outdoor courtyard. We select these environments to highlight our framework's ability to generalize in multiple scenarios. In each environment, we test multiple sentences across the four categories (see the Appendix for the full list of sentences). Sentences are primarily based on landmarks already present in the environment but in some cases, we add additional artifacts to enable more extensive testing.

Category	Count	Code(%)	OD(%)	WP(%)	Path&Exec(%)
Generic	22	100	81.82	68.18	68.18
Specific	19	89.47	89.47	78.95	73.68
Relational	44	70.45	56.82	56.82	56.82
Contextual	29	65.52	41.38	41.38	41.38
Total	114	78.07	63.16	58.77	57.89

Table 2: Summary of category-wise success rate for each of the four command types.

We find that we are able to successfully generate code for navigational plans using a variety of verbs e.g., walk, go, drive, run, etc by leveraging the rich vocabulary knowledge present in foundational models. In fact, we can even say phrases like "shashay to the stop sign" which will be interpreted as a "go to" style navigation command.

Scenes	Count	Code(%)	OD(%)	WP(%)	Path&Exec(%)
Theater	30	90	70	66.67	63.33
Lobby	29	65.52	48.28	44.83	44.83
Outdoor	24	87.5	79.17	70.83	70.83
Courtyard	31	70.97	58.06	54.84	54.84
Total	114	78.07	63.16	58.77	57.89

Table 3: Summary of scene-wise success rate for each of the four command types.

From Tables 2 and 3, we see that for generic objects our framework has a 100% success rate for code generation. In both tables *Code* represents the success of the code generation step, *OD* represents

the success of the object detector (GLIP), *WP* is the success of the 3D waypoint projection and *Path&Exec* is the success of the graph-based planning and navigation module. Our failures occurred in the object detection module where either the wrong object was detected or the object was not detected at all. However, since our framework is modular, the object detector can be substituted and as state-of-the-art vision detectors improve our framework will improve in turn. Additional failures occur in the waypoint projection step. These occurred for two reasons. In the first case, the vision system detected further out than the map resulting in projections at the edge of the map rather than the actual object. For the second case, projections missed the correct voxel on the map. This was either due to a ray "clipping" a closer object in the projection process. For longer projections (10–12m), the camera-to-lidar calibration caused enough error to make the projection inaccurate, which could be resolved through mean point clustering. We also note that we observed a 100% success rate in following all planned paths.

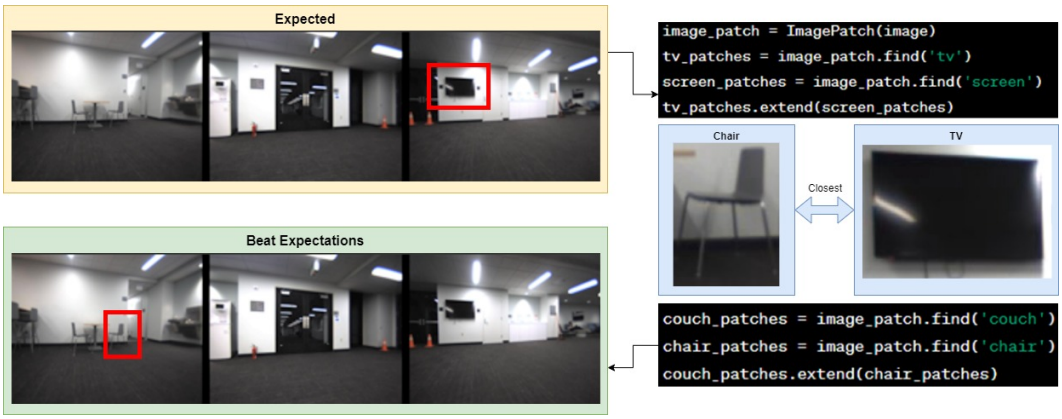


Figure 5: Results for the sentence "Find me somewhere to watch a movie."

Selected Qualitative Examples:

Leveraging the vast knowledge base encoded into LLMs enables a degree of spatial reasoning and the ability to infer navigation instructions based on the context of the sentence. For example, take the sentence "Find me something to help a firefighter" as shown in Figure 1 requires the robot to identify a list of objects that could help a firefighter and then look for them. From the snippet of the generated code, we see that the robot successfully looks for fire extinguishers and fire hydrants, both of which would aid a firefighter. Additionally, we are able to ask the robot to "Find something to clean up a mess" and it will find a mop or a broom.

What's more remarkable than simply interpreting correct navigation commands from context is our framework's ability to leverage foundational models to find better solutions to a task than we as humans would initially try and find. Take for example the sentence "Find me somewhere to watch a movie," we would expect the robot to go and find a tv or some kind of screen. Of course, the first object the robot looks for is the screen but more remarkably the robot also looks for a chair which is closest to the screen. We can see this in Figure 5 where the generated layer θ explicitly looks for "TVs, screens" and then looks for "chairs" and "couches."

As powerful as the contextual engine (GPT-3.5) is in our framework, it still has limitations when interpreting the physical world. In Figure 6 we tell the robot to "Run upstairs." and to no surprise, it will successfully generate code and plan up the stairs. However, when given the utterance "Go to the second floor." the following code is generated:

```
floor_patches = ImagePatch(image).find('floor')
floor_patches.sort(key=lambda x: x.vertical_center)
if len(floor_patches) < 2:
    return {'function': 'None', 'error': 'Image does not contain at least
        two floors.'}
second_floor_patch = floor_patches[1]
```

This code is nonsensical to the actual task because it's a literal interpretation of the sentence. Counting floors has nothing to do with going to the second story of a building. Examples like these highlight the need for additional methods of encoding 3D spatial reasoning into natural language based embodied navigation frameworks.

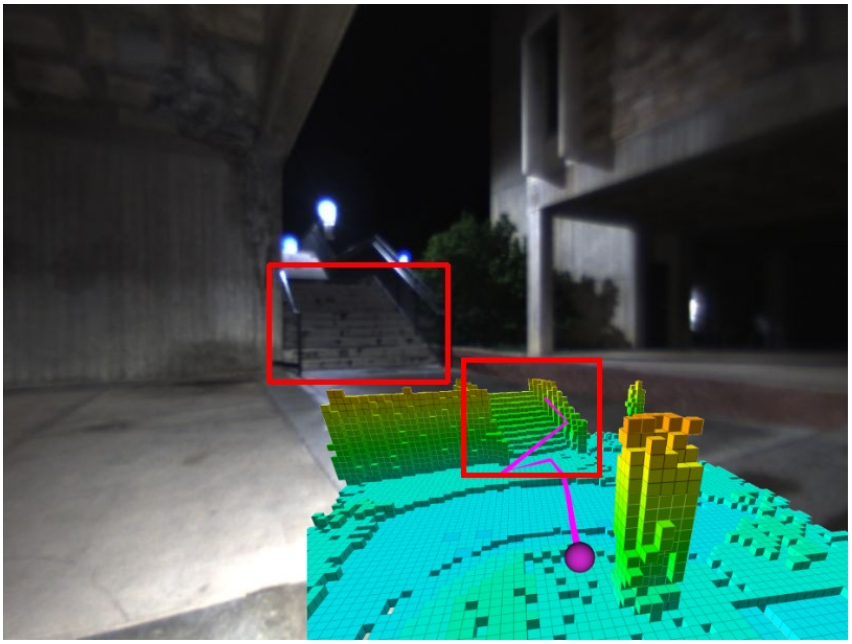


Figure 6: Scene used to tell the robot to "Run upstairs" and "Go to the second floor". The second sentence fails due to the code trying to "count floors"

5 Conclusion and Limitations

Our composable framework shows remarkable accuracy and performance in generating code that fits the navigation task. In controlled experiments, we achieve 90% accuracy across 50 different sentences. When taken to the real world the, model still performs exceptionally well with zero additional input. This is highlighted by a 70% execution accuracy in large outdoor environments.

While this work makes significant strides toward embodied navigation, we observe many opportunities for future effort. Namely, operating in only 2D space creates challenges for spatial relationships such as "behind" the robot or "to the right of." Additionally, we are limited by the visual range of the robot: therefore 3D projections need to be exponentially more accurate for both the smaller and further away an object is in the given input image. Finally, large multimodal foundational models are promising, but promise uncertain runtime performance given their lack of availability and scalability. Future work includes extending this framework to include more navigational inputs such as unbounded exploration in a direction or, having the robot explore until a condition is met. We also wish to include a module that operates on 3D maps and a module that supports sequential instructions.

References

[1] T. P. McNamara, J. K. Hardy, and S. C. Hirtle. Subjective hierarchies in spatial memory. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 15(2):211, 1989.

[2] DARPA. DARPA Subterranean Challenge, 2022. URL https://www.darpa.mil/program/darpa-subterranean-challenge. https://www.darpa.mil/program/darpa-subterranean-challenge.

[3] A. Agha, K. Otsu, B. Morrell, D. D. Fan, R. Thakker, A. Santamaria-Navarro, S.-K. Kim, A. Bouman, X. Lei, J. Edlund, et al. Nebula: Quest for robotic autonomy in challenging environments; team costar at the darpa subterranean challenge. *arXiv preprint arXiv:2103.11470*, 2021.

[4] H. Biggie, E. R. Rush, D. G. Riley, S. Ahmad, M. T. Ohradzansky, K. Harlow, M. J. Miles, D. Torres, S. McGuire, E. W. Frew, et al. Flexible supervised autonomy for exploration in subterranean environments. *arXiv preprint arXiv:2301.00771*, 2023.

[5] M. Tranzatto, T. Miki, M. Dharmadhikari, L. Bernreiter, M. Kulkarni, F. Mascarich, O. Andersson, S. Khattak, M. Hutter, R. Siegwart, et al. Cerberus in the darpa subterranean challenge. *Science Robotics*, 7(66):eabp9742, 2022.

[6] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong, et al. A survey of large language models. *arXiv preprint arXiv:2303.18223*, 2023.

[7] A. Glaese, N. McAleese, M. Trebacz, J. Aslanides, V. Firoiu, T. Ewalds, M. Rauh, L. Weidinger, M. Chadwick, P. Thacker, et al. Improving alignment of dialogue agents via targeted human judgements. *arXiv preprint arXiv:2209.14375*, 2022.

[8] M. Tsimpoukelli, J. L. Menick, S. Cabi, S. Eslami, O. Vinyals, and F. Hill. Multimodal few-shot learning with frozen language models. *Advances in Neural Information Processing Systems*, 34:200–212, 2021.

[9] P. Liu, W. Yuan, J. Fu, Z. Jiang, H. Hayashi, and G. Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35, 2023.

[10] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. d. O. Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[11] Y. Li, D. Choi, J. Chung, N. Kushman, J. Schrittwieser, R. Leblond, T. Eccles, J. Keeling, F. Gimeno, A. Dal Lago, et al. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, 2022.

[12] F. F. Xu, U. Alon, G. Neubig, and V. J. Hellendoorn. A systematic evaluation of large language models of code. In *Proceedings of the 6th ACM SIGPLAN International Symposium on Machine Programming*, pages 1–10, 2022.

[13] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in Neural Information Processing Systems*, 35:23716–23736, 2022.

[14] D. Surís, S. Menon, and C. Vondrick. Vipergpt: Visual inference via python execution for reasoning. *arXiv preprint arXiv:2303.08128*, 2023.

[15] L. H. Li, P. Zhang, H. Zhang, J. Yang, C. Li, Y. Zhong, L. Wang, L. Yuan, L. Zhang, J.-N. Hwang, et al. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10965–10975, 2022.

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

[17] T. Dang, M. Tranzatto, S. Khattak, F. Mascarich, K. Alexis, and M. Hutter. Graph-based subterranean exploration path planning using aerial and legged robots. *Journal of Field Robotics*, 37(8):1363–1388, 2020.

[18] S. Ahmad and J. S. Humbert. Efficient sampling-based planning for subterranean exploration. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7114–7121. IEEE, 2022.

[19] S. Tellex, T. Kollar, S. Dickerson, M. Walter, A. Banerjee, S. Teller, and N. 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.

[20] T. M. Howard, S. Tellex, and N. 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] T. Kollar, S. Tellex, D. Roy, and N. Roy. Toward understanding natural language directions. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, pages 259–266. IEEE, 2010.

[22] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu, et al. Palm-e: An embodied multimodal language model. *arXiv preprint arXiv:2303.03378*, 2023.

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

[24] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[25] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[26] A. Brohan, Y. Chebotar, C. Finn, K. Hausman, A. Herzog, D. Ho, J. Ibarz, A. Irpan, E. Jang, R. 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.

[27] M. S. Sajjadi, D. Duckworth, A. Mahendran, S. van Steenkiste, F. Pavetic, M. Lucic, L. J. Guibas, K. Greff, and T. Kipf. Object scene representation transformer. *Advances in Neural Information Processing Systems*, 35:9512–9524, 2022.

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

[29] J. Johnson, B. Hariharan, L. Van Der Maaten, J. Hoffman, L. Fei-Fei, C. Lawrence Zitnick, and R. Girshick. Inferring and executing programs for visual reasoning. In *Proceedings of the IEEE international conference on computer vision*, pages 2989–2998, 2017.

[30] R. Hu, J. Andreas, M. Rohrbach, T. Darrell, and K. Saenko. Learning to reason: End-to-end module networks for visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 804–813, 2017.

[31] K. Yi, J. Wu, C. Gan, A. Torralba, P. Kohli, and J. Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. *Advances in neural information processing systems*, 31, 2018.

[32] S. Antol, A. Agrawal, J. Lu, M. Mitchell, D. Batra, C. L. Zitnick, and D. Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[33] A. Madaan, S. Zhou, U. Alon, Y. Yang, and G. Neubig. Language models of code are few-shot commonsense learners. *arXiv preprint arXiv:2210.07128*, 2022.

[34] J. Li, D. Li, S. Savarese, and S. Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. *arXiv preprint arXiv:2301.12597*, 2023.

[35] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[36] J. Redmon, S. Divvala, R. Girshick, and A. 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.

[37] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[38] R. Ranftl, K. Lasinger, D. Hafner, K. Schindler, and V. Koltun. Towards robust monocular depth estimation: Mixing datasets for zero-shot cross-dataset transfer. *IEEE transactions on pattern analysis and machine intelligence*, 44(3):1623–1637, 2020.

[39] C. Zhao, Q. Sun, C. Zhang, Y. Tang, and F. Qian. Monocular depth estimation based on deep learning: An overview. *Science China Technological Sciences*, 63(9):1612–1627, 2020.

[40] S. F. Bhat, I. Alhashim, and P. Wonka. Adabins: Depth estimation using adaptive bins. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4009–4018, 2021.

[41] P. Vaithilingam, T. Zhang, and E. 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.

[42] E. Nijkamp, B. Pang, H. Hayashi, L. Tu, H. Wang, Y. Zhou, S. Savarese, and C. Xiong. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*, 2022.

[43] T. T. Mac, C. Copot, D. T. Tran, and R. De Keyser. Heuristic approaches in robot path planning: A survey. *Robotics and Autonomous Systems*, 86:13–28, 2016.

[44] H.-y. Zhang, W.-m. Lin, and A.-x. Chen. Path planning for the mobile robot: A review. *Symmetry*, 10(10):450, 2018.

[45] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller. Anytime motion planning using the rrt. In *2011 IEEE international conference on robotics and automation*, pages 1478–1483. IEEE, 2011.

[46] M. Pivtoraiko, R. A. Knepper, and A. Kelly. Differentially constrained mobile robot motion planning in state lattices. *Journal of Field Robotics*, 26(3):308–333, 2009.

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

[48] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song. Clip on wheels: Zero-shot object navigation as object localization and exploration. *arXiv preprint arXiv:2203.10421*, 2022.

[49] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pages 8748–8763. PMLR, 2021.

[50] B. 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] D. Shah, B. Osiński, S. 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.

[52] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine. Ving: Learning open-world navigation with visual goals. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 13215–13222. IEEE, 2021.

[53] C. Huang, O. Mees, A. Zeng, and W. Burgard. Visual language maps for robot navigation. *arXiv preprint arXiv:2210.05714*, 2022.

[54] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

[55] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[56] J. Andreas, M. Rohrbach, T. Darrell, and D. Klein. Neural module networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 39–48, 2016.

[57] T. Gupta and A. 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.

[58] J. Mao, C. Gan, P. Kohli, J. B. Tenenbaum, and J. Wu. The neuro-symbolic concept learner: Interpreting scenes, words, and sentences from natural supervision. *arXiv preprint arXiv:1904.12584*, 2019.

[59] P. Krüsi, P. Furgale, M. Bosse, and R. Siegwart. Driving on point clouds: Motion planning, trajectory optimization, and terrain assessment in generic nonplanar environments. *Journal of Field Robotics*, 34(5):940–984, 2017.

[60] A. Hornung, K. M. Wurm, M. Bennewitz, C. Stachniss, and W. Burgard. Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34:189–206, 2013.

[61] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus. Lio-sam: Tightly-coupled lidar inertial odometry via smoothing and mapping. In *2020 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 5135–5142. IEEE, 2020.