Verifiably Following Complex Robot Instructions with Foundation Models

Benedict Quartey[†*], Eric Rosen[*], Stefanie Tellex, George Konidaris
Department of Computer Science, Brown University

***Abstract*— When instructing robots, users want to flexibly express constraints, refer to arbitrary landmarks, and verify robot behavior, while robots must disambiguate instructions into specifications and ground instruction referents in the real world. To address this problem, we propose Language Instruction grounding for Motion Planning (LIMP), an approach that enables robots to verifiably follow complex, open-ended instructions in real-world environments without prebuilt semantic maps. LIMP constructs a symbolic instruction representation that reveals the robot's alignment with an instructor's intended motives and affords the synthesis of correct-by-construction robot behaviors. We conduct a large-scale evaluation of LIMP on 150 instructions across five real-world environments, demonstrating its versatility and ease of deployment in diverse, unstructured domains. LIMP performs comparably to state-of-the-art baselines on standard open-vocabulary tasks and additionally achieves a 79% success rate on complex spatiotemporal instructions, significantly outperforming baselines that only reach 38%.** [1]

I. Introduction

Robots need a rich understanding of natural language to be instructable by non-experts in unstructured environments. People, on the other hand, need to be able to verify that a robot has understood a given instruction and will act appropriately. Achieving these objectives, however, is challenging as natural language instructions often feature ambiguous phrasing, intricate spatiotemporal constraints, and unique referents. To illustrate, consider the instruction shown in Figure 1: *"Bring the green plush toy to the whiteboard in front of it, watch out for the robot in front of the toy"*. Solving such a task requires a robot to ground open-vocabulary referents, follow temporal constraints, and disambiguate objects using spatial descriptions. Foundation models [1], [2] offer a path to achieving such complex long-horizon goals; however, existing approaches for robot instruction following have largely focused on navigation [3], [4], [5], [6], [7]. These methods, broadly classified under object goal navigation [8], enable navigation to instances of an object category but are limited in their ability to localize spatial references and disambiguate object instances based on descriptive language. Other works [9], [10], [11] extend instruction following to mobile manipulation but are limited to tasks with simple temporal constraints expressed in unambiguous language. Moreover, existing efforts typically rely on Large Language Models (LLMs) as complete planners, bypassing intermediate symbolic representations that could provide verification of correctness before execution. Alternative approaches leveraging code-writing LLMs [5], [6], [12] are susceptible to errors in generated code, which may lead to unsafe robot behaviors. Mapping natural language to specification languages like temporal logic [13] provides a robust framework for language disambiguation, handling complex temporal constraints, and behavior verification. However, prior works along this line require prebuilt semantic maps with discrete sets of prespecified referents/landmarks from which instructions can be constructed [7], [14], [15].

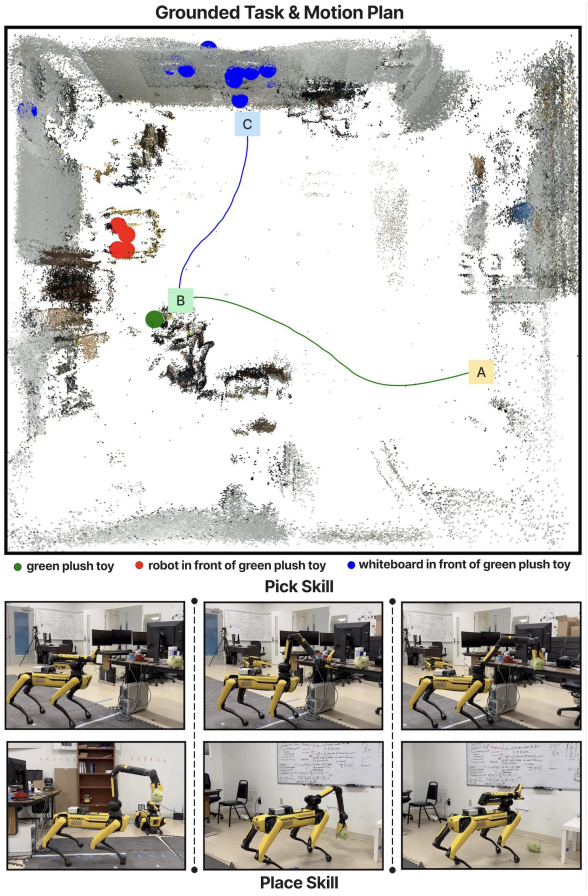

Fig. 1: Our approach executing the instruction *"Bring the green plush toy to the whiteboard in front of it, watch out for the robot in front of the toy"*. The robot dynamically detects and grounds open-vocabulary referents with spatial constraints to construct an instruction-specific semantic map, then synthesizes a task and motion plan to solve the task. In this example, the robot navigates from its start location (yellow, A), to the green plush toy (green, B), executes a pick skill then navigates to the whiteboard (blue, C), and executes a place skill. Note that the robot has no prior semantic knowledge of the environment.

We propose *Language Instruction grounding for Motion Planning (LIMP)*, a method that leverages foundation models

[*]Equal Contribution
[†]Corresponding Author (*Email*: benedict_quartey@brown.edu)
[1]See supplementary materials and demo videos at robotlimp.github.io

and temporal logics to dynamically generate instruction-conditioned semantic maps that enable robots to construct verifiable controllers for following navigation and mobile manipulation instructions with open vocabulary referents and complex spatiotemporal constraints. In a novel environment, LIMP constructs a 3D map via SLAM, then uses LLMs to translate complex natural language instructions into temporal logic specifications with a novel composable syntax for referent disambiguation. Instruction referents are detected and grounded using vision-language models (VLMs) and spatial reasoning. Finally, a task and motion plan is synthesized to guide the robot through the required subgoals, as shown in Figure 1. In summary, we make the following contributions: (**1**) A modular framework that translates expressive natural language instructions into temporal logic, grounds instruction referents, and executes commands via Task and Motion Planning (TAMP). (**2**) A spatial grounding method for detecting and localizing open vocabulary objects with spatial constraints in 3D metric maps. (**3**) A TAMP algorithm that localizes regions of interest (goal/avoidance zones) and synthesizes constraint-satisfying motion plans for long-horizon tasks.

II. Background and Related Works

We briefly highlight the most relevant works in visual scene understanding [10], natural language instruction following [7], [16], and task and motion planning [17], and provide a comprehensive review in our supplementary materials. NLMap [10] grounds open-vocabulary language queries to spatial locations using pre-trained VLMs. While effective for describing individual objects, it cannot handle instructions involving complex constraints between multiple objects due to the lack of object relationship modeling. LIMP addresses this with a novel spatial grounding module that resolves spatial relationships and leverages task and motion planners to satisfy these constraints. Lang2LTL [7] is a multi-stage, LLM-based approach that uses entity extraction and replacement to translate language instructions into temporal logic. Its extension [16] incorporates VLMs and semantic information (via text embeddings) to ground referents. These works require prebuilt semantic maps/databases describing landmarks to ground symbols, whereas our approach dynamically generates landmarks based on open-vocabulary instructions. Action-Oriented Semantic Maps (AOSMs) [17] augment semantic maps with models indicating where robots can perform manipulation skills, integrating with TAMP solvers for mobile manipulation. LIMP similarly provides a TAMP-compatible spatial representation but supports open-vocabulary tasks, whereas AOSMs remain constrained to a fixed set of goals once generated.

A. Linear Temporal Logic

LIMP translates natural language instructions into temporal logic specifications for verifiable task and motion planning. While compatible with various specification languages and planning frameworks, we choose Linear Temporal Logic (LTL) [18] for its proven expressivity in representing complex robot mission requirements [19]. LTL defines temporal properties using atomic propositions, logical operators—negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$), implication ($\rightarrow$)—and temporal operators: next ($\mathcal{X}$), until ($\mathcal{U}$), globally ($\mathcal{G}$), and finally ($\mathcal{F}$). Despite its expressivity, LTL has been underutilized due to the expert knowledge required to construct specifications, however recent works have seen significant success directly translating natural language into LTL [7], [20], [14], [21], [22], [23].

Behavior Verification: Expressing instructions as temporal logic specifications allows us to verify the correctness of generated plans a priori. However, instead of explicit verification methods such as model checking, we leverage insights from prior works [24] and directly use specifications to synthesize plans that are *correct-by-construction* [25], [26].

III. Problem Definition

Given a natural language instruction l, our goal is to synthesize and sequence navigation and manipulation behaviors to produce a policy that satisfies the temporal and spatial constraints in l. Spatial constraints determine task success based on the sequence of robot poses traversed during execution; temporal constraints determine the sequencing of these spatial constraints as a function of task progression. We assume a robot with an RGB-D camera has already navigated a space, capturing images and camera poses. From this data, we build a metric map m (e.g., point cloud, 3D voxel grid) of the environment, defining the space of possible $SE(3)$ poses P and enabling robot localization (i.e., estimating $p_{\text{robot}} \in P$). Unlike previous work leveraging temporal logic [7], we do not assume access to a semantic map with prespecified object locations or predicates. Instead, we leverage two foundation models: a task-agnostic vision-language model σ that, given an image and text, provides bounding boxes or segmentations based on the text; and an auto-regressive large language model ψ that samples likely language tokens based on a history of tokens.

Navigation: Navigation is formalized as an object-goal oriented continuous path planning problem, where the goal is to generate paths to a goal pose set $P_{goals} \subset P$ while staying in feasible regions ($P_{feasible} \subset P$) and avoiding infeasible regions ($P_{infeasible} = P^C_{feasible}$). Infeasible regions include environment obstacles as well as dynamically determined semantic regions that violate constraints in the instruction l.

Manipulation: We formalize manipulation behaviours as options [27] parameterized by objects. Consider an object parameter θ that parameterizes an option $O_\theta = (I_\theta, \pi_\theta, \beta_\theta)$, the initiation set, policy, and termination condition are functions of both the robot pose P and θ. The initiation set I_θ denotes the global reference frame robot positions and object-centric attributes—such as object size—that determine if the option policy π_θ can be executed on the object θ. To execute a manipulation skill on an object, an object-goal navigation behavior must first be executed to bring the robot into proximity with the object. We assume access to a library of these manipulation skills and demonstrate our approach

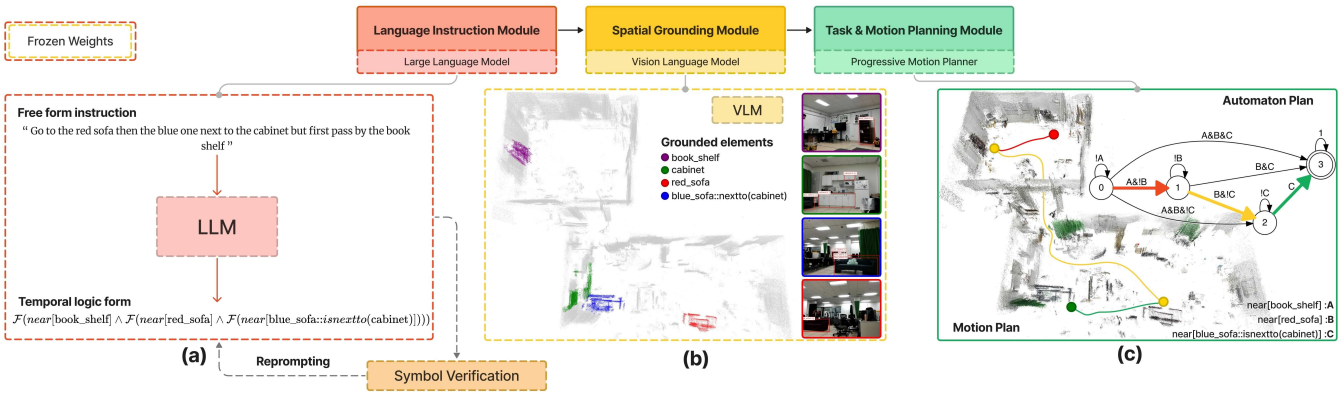

Fig. 2: **[A]** LIMP translates natural language instructions into temporal logic expressions, where open-vocabulary referents are applied to predicates that correspond to robot skills—note the context-aware resolution of the phrase "blue one" to the referent "blue_sofa". **[B]** Vision-language models detect referents, while spatial reasoning disambiguates referent instances to generate a 3D semantic map that localizes instruction-specific referents. **[C]** Finally, the temporal logic expression is compiled into a finite-state automaton, which a task and motion planner uses with dynamically-generated task progression semantic maps to progressively identify goals and constraints in the environment, and generate a plan that satisfies the high-level task specification.

on multi-object goal navigation and open-vocabulary mobile pick-and-place [9], [28].

IV. Language Instruction Grounding for Motion Planning

LIMP interprets expressive natural language instructions to generate instruction-conditioned semantic maps, enabling robots to verifiably solve long-horizon tasks with complex spatiotemporal constraints (Figure 2). We briefly describe our modular approach in this section and present comprehensive implementations details in our supplementary materials.

A. Language Instruction Module

In this module, we leverage a large language model ψ to translate a natural language instruction l into a linear temporal logic specification φ_l with a novel composable syntax for referent disambiguation. We achieve this through a two-stage in-context learning strategy. The first stage prompts ψ to translate l into a conventional LTL formula ϕ_l where propositions refer to open-vocabulary objects. The second stage takes l and ϕ_l as input and prompts ψ to generate a new formula φ_l with predicate functions corresponding to parameterized robot skills.

We define three predicate functions—**near**, **pick**, and **release**—for the primitive navigation and manipulation skills required for multi-object goal navigation and mobile pick-and-place. Predicate functions in φ_l are parameterized by *Composable Referent Descriptors* (CRDs), our novel propositional expressions representing specific referent instances by chaining comparators that encode descriptive spatial information. For example, the instruction *"the yellow cabinet above the fridge that is next to the stove"* can be represented with the CRD:

$$\text{yellow\_cabinet} :: \text{isabove}(\text{fridge} :: \text{isnextto}(\text{stove})). \quad (1)$$

This specifies that there is a fridge *next to* a stove, and the desired yellow cabinet is *above* that fridge. CRDs are constructed from a set of 3D spatial comparators [29] defined in our prompting strategy.

Input Instruction : " Bring the green plush toy to the whiteboard in front of it, watch out for the robot in front of the toy"

Conventional LTL : $\underset{\phi_l}{\mathcal{F}(\text{green\_plush\_toy} \wedge \mathcal{F}(\text{whiteboard} \wedge \neg\text{robot}))}$

Our LTL Syntax : $\underset{\varphi_l}{\mathcal{F}(A \wedge \mathcal{F}(B \wedge \mathcal{F}(C \wedge \neg D \wedge \mathcal{F}E)))}$

```
A: near[green_plush_toy]
B: pick[green_plush_toy]
C: near[whiteboard::isinfrontof(green_plush_toy)]
D: near[robot::isinfrontof(green_plush_toy)]
E: release[green_plush_toy, whiteboard::infrontof(green_plush_toy)]
```

Fig. 3: An instruction is first translated into a conventional LTL formula ϕ_l that loosely captures the desired temporal occurrence of referent objects, then into our LTL syntax φ_l with predicate functions that temporally chain required robot skills parameterized by composable referent descriptors.

Unlike recent works [9], [11], our approach does not require specific phrasing or keywords and can handle instructions with arbitrary complexity and ambiguity. The LLM ψ directly samples the entire LTL formula φ_l with predicate functions parameterized by CRDs using appropriate spatial comparators based on the instruction's context. Figure 3 illustrates the result of our two-stage prompting strategy.

LLM Verification: Verifying the LTL formula φ_l sampled from the LLM is crucial as errors in referent extraction and temporal task structure affects instruction following accuracy. Our symbol verification node (Figure 2) leverages LTL properties to provide high-level human-in-the-loop verification of extracted instruction referents and temporal task structure. Recent work [30] provides ISO 61508 [31] safety guarantees in robot task execution by translating safety constraints from natural language to LTL formulas, which are verified by human experts and used to enforce robot behavior. Similarly, we rely on human verification to ensure the translated formula φ_l is correct. Our symbol verification node implements an interactive dialog system that presents users with the extracted referent CRDs and implied task structure, and reprompts the LLM based on user corrections to obtain new formulas. Unlike prior work [30], we eliminate the need for experts by directly translating the task structure—encoded in the LTL formulas's equivalent automaton—back into English

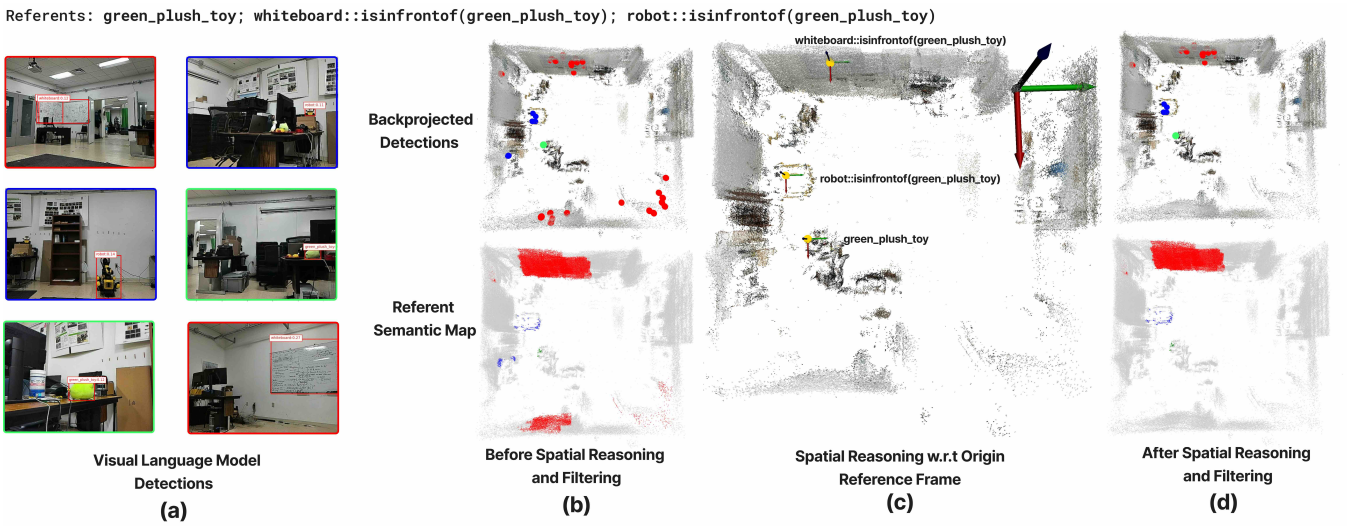

Fig. 4: **[A]** Our spatial grounding module leverages a VLM to detect all referent occurrences from prior observations of the environment. **[B]** An initial semantic map with all detected referent instances is generated by backprojecting pixels in segmented referent masks unto the 3D map. **[C]** Each referent's spatial comparators is resolved with respect to the origin coordinate frame of reference. [D] Failing instances are filtered out to obtain a Referent Semantic Map (RSM) that localizes the exact referent instances described in the instruction.

statements via a simple deterministic translation scheme. In our experiments (Tables I and II), we find that even without human verification and reprompting, the initial formulas sampled by our language understanding module impressively encode the correct referents and temporal task structure.

B. Spatial Grounding Module

This module detects and localizes specific object instances referenced in an instruction. From the translated LTL formulas, we extract composable referent descriptors (CRDs) and use vision-language models OWL-ViT [32] and SAM [33] to detect and segment all referent occurrences from the robot's prior observations of the environment. We backproject pixels in these segmentation masks onto our 3D map, creating an initial semantic map of all instruction object instances. From the example in Figure 3, occurrences of *green_plush_toy*, *whiteboard*, and *robot* are detected, segmented, and backprojected onto the map (Figure 4[a&b]).

To obtain the specific object instances described in the instruction, we resolve the 3D spatial comparators in each referent's CRD—recall that CRDs are propositional expressions and can be evaluated as true or false. We define eight spatial comparators (*isbetween*, *isabove*, *isbelow*, *isleftof*, *isrightof*, *isnextto*, *isinfrontof*, *isbehind*) to reason about spatial relationships based on backprojected 3D positions. Since all backprojected positions are relative to an origin coordinate system, our spatial comparators are resolved from the perspective of this origin position as shown in Figure 4[c]. This type of relative frame of reference (FoR) when describing spatial relationships between objects, in contrast to an absolute or intrinsic FoR, is dominant in English [34], and is a logical choice for our work.

Using the 3D position of each referent's center mask pixel as its representative position, we resolve a given referent with a spatial description by applying the appropriate spatial comparator to all detected pairs of the desired referent and comparison landmark objects. This filtering process yields a Referent Semantic Map (RSM) that localizes specific object instances described in the instruction as shown in Figure 4[d].

VLM Verification: Potential misclassifications from object detector VLMs is the main source of error in this module. We do not address interactively correcting VLM misclassifications as that is out of the scope of this work, but we provide 3D visualization tools that enable users to visually inspect and verify that constructed referent semantic maps correctly localize referents.

C. Task and Motion Planning Module

Finally, our TAMP module synthesizes and sequences navigation and manipulation behaviors to produce a plan that satisfies the temporal and spatial constraints expressed in the given instruction.

Progressive Motion Planner (PMP): Our TAMP algorithm compiles the LTL formula with parameterized robot skills into an equivalent finite-state automaton (Figure 5[a]) to generate a verifiably correct task and motion plan. A path from the initial to the accepting state in this automaton is a high-level task plan that interleaves navigation and manipulation objectives required to satisfy the instruction. We select such a path with a simple strategy that incrementally selects the next progression state until the accepting state is reached, ensuring the plan obeys all temporal subgoal objectives. As shown in Figure 5[a], automaton states are connected by transition edges representing the logical expressions required for transitions. For each transition, our algorithm executes the necessary low-level behaviors: for manipulation subgoals, it executes the appropriate parameterized skill; for navigation subgoals, it dynamically generates Task Progression Semantic Maps (TPSMs) to localize goal and constraint regions and performs continuous path planning using the Fast Marching Tree algorithm (FMT*) [35].

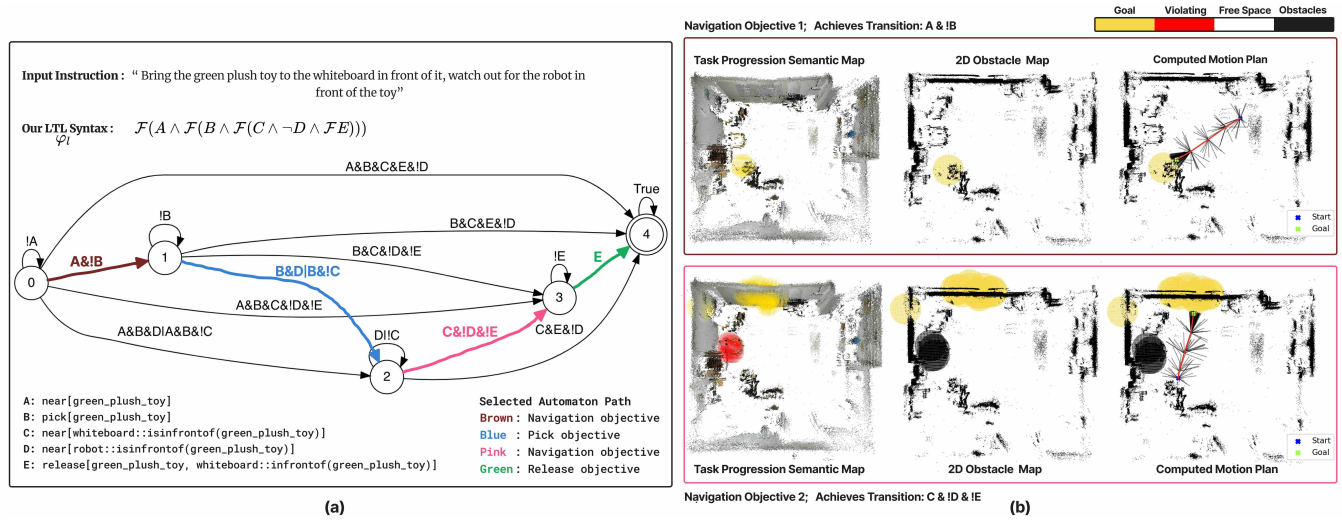

Fig. 5: **[A]** A given instruction translated into our LTL syntax φ_l can be compiled into an equivalent finite-state automaton that captures the temporal constraints of the task. A path through this automaton is selected with a strategy that incrementally picks the next progression state from the initial state to the accepting state. The robot then executes the manipulation options and navigation behaviors dictated by this high-level task plan. **[B]** To execute navigation objectives our approach generates a task progression semantic map (TPSM) that augments the environment with state transition constraints, localizing goal (yellow) and avoidance (red) regions. Generated TPSMs are converted into 2D obstacle maps for constraint-aware continuous path planning.

Task Progression Semantic Maps (TPSM): A TPSM augments a 3D scene with navigation constraints specified by logical state transition expressions, enabling goal-directed, constraint-aware navigation. Regions of interest in a TPSM are defined using a nearness threshold specifying proximity to an object. This threshold can be set globally or included in the language instruction module's prompting strategy, allowing an LLM to infer its value based on the instruction. Like our spatial grounding module, TPSMs are agnostic to temporal logic representations and can be used with various planning approaches for semantic constraint-aware motion planning. We primarily evaluate our approach on a ground mobile robot, hence we transform 3D TPSMs into 2D geometric obstacle maps, where constraint regions are treated as obstacles (Figure 5[b]). However, our approach is robot-agnostic and supports direct planning in 3D TPSMs for appropriate embodiments like drones.

V. Evaluation

Our evaluations test the hypothesis that translating natural language instructions into LTL expressions and dynamically generating semantic maps enables robots to accurately interpret and execute instructions in large-scale environments without prior training. We focus on three key questions: (1) Can our language instruction module interpret complex, ambiguous instructions? (2) Can our spatial grounding module resolve specific object instances described in instructions? (3) Can our TAMP algorithm generate constraint-satisfying plans?

To answer these questions, we compare LIMP with two baselines: an LLM task planner (NLMap-Saycan [10]) and an LLM code-writing planner (Code-as-Policies [12]), representing state-of-the-art approaches for language-conditioned, open-ended robot task execution. Both baselines use the same LLM (GPT-4-0613), prompting structure, and in-context learning examples as our language instruction module. In Code-as-Policies, in-context examples are converted into language model-generated program (LMP) snippets [12]. To ensure competitive performance, we integrate our CRD syntax, spatial grounding module, and low-level robot control into these baselines, allowing them to query object positions, use our FMT* path planner, and execute manipulation skills.

We also evaluate ablations of our two-stage language instruction module due to its importance in instruction following. In our full approach, the first stage prompts an LLM to generate a conventional LTL formula ϕ_l from instruction l by dynamically selecting relevant in-context examples from a standard dataset [14] based on cosine similarity. Our first ablation selects in-context examples randomly; and the second ablation skips this stage entirely, directly sampling our LTL syntax with parameterized robot skills φ_l from l.

We conduct a large-scale evaluation across five real-world environments on a diverse task set of 150 instructions from multiple prior works [10], [11], [7]. This task set consists of 24 tasks with fine-grained object descriptions (NLMD), 25 tasks with complex language (NLMC), 25 tasks with simple structured phrasing (OKRB), 37 tasks with complex temporal structures (CT) and 39 tasks with descriptive spatial constraints and temporal structures (CST). Below are examples from each task category illustrating the variety in complexity:

```
NLMD: Put the brown multigrain chip bag in the woven basket
NLMC: I like fruits, can you put something I would like on the yellow sofa for me
OKRB: Move the soda can to the box
CT: Visit the purple door elevator, then go to the front desk and then go to the kitchen table, in addition you can never go to the elevator once you have seen the front desk
CST: I have a white cabinet, a green toy, a bookshelf and a red chair around here somewhere. Take the second item I mentioned from between the first item and the third. Bring it the cabinet but avoid the last item at all costs.
```

To evaluate instruction understanding, we introduce performance metrics: **referent resolution accuracy**, **avoidance**

TABLE I: Performance comparison of one-shot instruction understanding and spatial resolution.

Approach	Referent Resolution Accuracy (Average WER) ↓	Avoidance Constraint Resolution Accuracy (Average WER) ↓	Spatial Relationship Resolution Accuracy (Average WER) ↓
NLMap-Saycan	0.09	0.12	0.05
Code-as-Policies	0.22	0.24	0.06
Limp Single Stage Prompting	0.09	0.11	0.05
Limp Two Stage Prompting [Random Embedding]	0.08	0.11	0.03
Limp Two Stage Prompting [Similar Embedding]	**0.07**	**0.04**	**0.03**

TABLE II: Performance comparison of one-shot temporal alignment and plan success rate.

Approach	Temporal Alignment Accuracy (% of instructions) ↑					Planning Success Rate (% of instructions) ↑				
	NLMD	NLMC	OKRB	CT	CST	NLMD	NLMC	OKRB	CT	CST
NLMap-Saycan	88%	**96%**	100%	32%	41%	75%	**96%**	100%	35%	38%
Code-as-Policies	58%	68%	100%	35%	38%	46%	68%	100%	38%	38%
Limp Single Stage Prompting	79%	64%	100%	68%	74%	63%	60%	100%	62%	62%
Limp Two Stage Prompting [Random Embedding]	83%	68%	100%	76%	85%	79%	68%	100%	57%	72%
Limp Two Stage Prompting [Similar Embedding]	**88%**	80%	**100%**	**76%**	**92%**	**79%**	76%	**100%**	**65%**	**79%**

constraint resolution accuracy, and **spatial relationship resolution accuracy**. These metrics utilize the word error rate (WER), widely used in speech recognition to quantify the difference between a reference and a hypothesis transcription by computing the minimal number of substitutions, deletions, and insertions needed to transform the hypothesis into the reference. WER is calculated as $WER = \frac{S+D+I}{N}$, where S is substitutions, D deletions, I insertions, and N is the total number of words in the reference. In our work:

- **Referent resolution accuracy** compares extracted referents in the generated LTL formula to ground truth referents.
- **Avoidance constraint resolution accuracy** compares referents to avoid in the LTL formula (denoted by the unary negation operator) against ground truth avoidance referents.
- **Spatial relationship resolution accuracy** compares generated Composable Referent Descriptors (CRDs) in the LTL specification with ground truth CRD expressions.

We also define **temporal alignment accuracy** and **planning success rate**. A plan is temporally aligned if the sequence of subgoals matches the instructor's intention, and successful if it satisfies all spatial and temporal constraints specified in the instruction. Achieving a high plan success rate is challenging, requiring accurate referent and avoidance constraint resolution, spatial grounding, and temporal alignment. We report average word error rates for each baseline in Table I and the percentage of successful and temporally accurate plans in Table II.

VI. Discussion

Beyond the verification benefits of symbolic planning, LIMP outperforms baselines in most task sets, notably in complex temporal planning and constraint avoidance. While NLMap-Saycan and Code-as-Policies effectively generate sequential subgoals, they struggle with strict temporal constraints—for example, avoiding a specific referent while approaching another. Our approach ensures each robot step adheres to constraints while achieving subgoals, explaining LIMP's superior performance on CT and CST tasks. As shown in Table II, LIMP underperforms only against NLMap-Saycan in the NLMC task category. This task set, introduced in the same paper as the baseline [10] (which outperforms LIMP), includes instructions with implicit details such as: "I like fruits, can you put something I would like on the yellow sofa for me." NLMap-Saycan is better suited to infer and generate plans with possible fruit options, whereas our few-shot LTL translation process is not designed for this.

VII. Limitations and Conclusion

Although LIMP is capable of interpreting non-finite instructions into LTL formulas, our planner is currently limited to processing co-safe formulas, which handle only finite sequences. The accuracy of spatial grounding relies on the performance of vision-language models (VLMs) for object recognition meaning any shortcomings in these systems can negatively impact results. Additionally, LIMP assumes a static environment between mapping and execution, making it not responsive to dynamic changes—an area we aim to address with future work on editable scene representations. Our Progressive Motion Planning algorithm is complete but does not guarantee optimality; however, our framework can be used with existing TAMP planners to enhance efficiency.

Foundation models hold significant promise for advancing the next generation of autonomous robots. Our results suggest that combining these models—LLMs for language and VLMs for vision—with established methods for safety, explainability, and verifiable behavior synthesis can lead to more reliable and capable robotic systems.

Acknowledgement

This work was supported by the Office of Naval Research (ONR) under REPRISM MURI N000142412603 and ONR #N00014-22-1-2592, as well as the National Science Foundation (NSF) via grant #1955361. Partial funding was also provided by The Robotics and AI Institute.

References

[1] Y. Hu, Q. Xie, V. Jain, J. Francis, J. Patrikar, N. Keetha, S. Kim, Y. Xie, T. Zhang, H.-S. Fang, S. Zhao, S. Omidshafiei, D.-K. Kim, A.-a. Agha-mohammadi, K. Sycara, M. Johnson-Roberson, D. Batra, X. Wang, S. Scherer, C. Wang, Z. Kira, F. Xia, and Y. Bisk, "Toward General-Purpose Robots via Foundation Models: A Survey and Meta-Analysis," Oct. 2024, arXiv:2312.08782 [cs]. [Online]. Available: http://arxiv.org/abs/2312.08782

[2] R. Firoozi, J. Tucker, S. Tian, A. Majumdar, J. Sun, W. Liu, Y. Zhu, S. Song, A. Kapoor, K. Hausman, B. Ichter, D. Driess, J. Wu, C. Lu, and M. Schwager, "Foundation models in robotics: Applications, challenges, and the future," *The International Journal of Robotics Research*, p. 02783649241281508, Sept. 2024, publisher: SAGE Publications Ltd STM. [Online]. Available: https://doi.org/10.1177/02783649241281508

[3] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song, "CoWs on Pasture: Baselines and Benchmarks for Language-Driven Zero-Shot Object Navigation," in *2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE, June 2023, pp. 23 171–23 181. [Online]. Available: https://ieeexplore.ieee.org/document/10203853/

[4] D. Shah, B. Osiński, B. Ichter, and S. Levine, "LM-Nav: Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action," in *Proceedings of The 6th Conference on Robot Learning*. PMLR, Mar. 2023, pp. 492–504, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v205/shah23b.html

[5] C. Huang, O. Mees, A. Zeng, and W. Burgard, "Visual Language Maps for Robot Navigation," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 10 608–10 615. [Online]. Available: https://ieeexplore.ieee.org/document/10160969

[6] ——, "Audio Visual Language Maps for Robot Navigation," in *Experimental Robotics*, M. H. Ang Jr and O. Khatib, Eds. Cham: Springer Nature Switzerland, 2024, pp. 105–117.

[7] J. X. Liu, Z. Yang, I. Idrees, S. Liang, B. Schornstein, S. Tellex, and A. Shah, "Grounding Complex Natural Language Commands for Temporal Tasks in Unseen Environments," in *Proceedings of The 7th Conference on Robot Learning*. PMLR, Dec. 2023, pp. 1084–1110, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v229/liu23d.html

[8] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva, and A. R. Zamir, "On Evaluation of Embodied Navigation Agents," July 2018, arXiv:1807.06757 [cs]. [Online]. Available: http://arxiv.org/abs/1807.06757

[9] S. Yenamandra, A. Ramachandran, K. Yadav, A. S. Wang, M. Khanna, T. Gervet, T.-Y. Yang, V. Jain, A. Clegg, J. M. Turner, Z. Kira, M. Savva, A. X. Chang, D. S. Chaplot, D. Batra, R. Mottaghi, Y. Bisk, and C. Paxton, "HomeRobot: Open-Vocabulary Mobile Manipulation," in *Proceedings of The 7th Conference on Robot Learning*. PMLR, Dec. 2023, pp. 1975–2011, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v229/yenamandra23a.html

[10] B. Chen, F. Xia, B. Ichter, K. Rao, K. Gopalakrishnan, M. S. Ryoo, A. Stone, and D. Kappler, "Open-vocabulary Queryable Scene Representations for Real World Planning," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 11 509–11 522. [Online]. Available: https://ieeexplore.ieee.org/document/10161534

[11] P. Liu, Y. Orru, J. Vakil, C. Paxton, N. Shafiullah, and L. Pinto, "Demonstrating OK-Robot: What Really Matters in Integrating Open-Knowledge Models for Robotics," in *Robotics: Science and Systems*. Robotics: Science and Systems Foundation, July 2024. [Online]. Available: http://www.roboticsproceedings.org/rss20/p091.pdf

[12] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as Policies: Language Model Programs for Embodied Control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 9493–9500. [Online]. Available: https://ieeexplore.ieee.org/document/10160591

[13] E. A. Emerson, "Temporal and Modal Logic," in *Handbook of Theoretical Computer Science, Volume B: Formal Models and Semantics*, J. v. Leeuwen, Ed. Elsevier and MIT Press, 1990, pp. 995–1072. [Online]. Available: https://doi.org/10.1016/b978-0-444-88074-1.50021-4

[14] J. Pan, G. Chou, and D. Berenson, "Data-Efficient Learning of Natural Language to Linear Temporal Logic Translators for Robot Task Specification," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*, May 2023, pp. 11 554–11 561. [Online]. Available: https://ieeexplore.ieee.org/document/10161125

[15] B. Quartey, A. Shah, and G. Konidaris, "Exploiting Contextual Structure to Generate Useful Auxiliary Tasks," in *NeurIPS 2023 Workshop on Generalization in Planning*, vol. abs/2303.05038, 2023, arXiv: 2303.05038. [Online]. Available: https://doi.org/10.48550/arXiv.2303.05038

[16] J. X. Liu, A. Shah, G. Konidaris, S. Tellex, and D. Paulius, "Lang2LTL-2: Grounding Spatiotemporal Navigation Commands Using Large Language and Vision-Language Models," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2024, pp. 2325–2332, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/document/10802696

[17] E. Rosen, S. James, S. Orozco, V. Gupta, M. Merlin, S. Tellex, and G. Konidaris, "Synthesizing Navigation Abstractions for Planning with Portable Manipulation Skills," in *Proceedings of The 7th Conference on Robot Learning*. PMLR, Dec. 2023, pp. 2278–2287, iSSN: 2640-3498. [Online]. Available: https://proceedings.mlr.press/v229/rosen23a.html

[18] A. Pnueli, "The temporal logic of programs," in *18th Annual Symposium on Foundations of Computer Science (sfcs 1977)*, Oct. 1977, pp. 46–57, iSSN: 0272-5428. [Online]. Available: https://ieeexplore.ieee.org/document/4567924

[19] C. Menghi, C. Tsigkanos, P. Pelliccione, C. Ghezzi, and T. Berger, " Specification Patterns for Robotic Missions ," *IEEE Transactions on Software Engineering*, vol. 47, no. 10, pp. 2208–2224, Oct. 2021. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TSE.2019.2945329

[20] M. Berg, D. Bayazit, R. Mathew, A. Rotter-Aboyoun, E. Pavlick, and S. Tellex, "Grounding Language to Landmarks in Arbitrary Outdoor Environments," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*, May 2020, pp. 208–215, iSSN: 2577-087X. [Online]. Available: https://ieeexplore.ieee.org/document/9197068

[21] M. Cosler, C. Hahn, D. Mendoza, F. Schmitt, and C. Trippel, "nl2spec: Interactively Translating Unstructured Natural Language to Temporal Logics with Large Language Models," in *Computer Aided Verification*, C. Enea and A. Lal, Eds. Cham: Springer Nature Switzerland, 2023, pp. 383–396.

[22] F. Fuggitti and T. Chakraborti, "NL2LTL – a Python Package for Converting Natural Language (NL) Instructions to Linear Temporal Logic (LTL) Formulas," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 13, pp. 16 428–16 430, 2023, number: 13. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/27068

[23] Y. Chen, R. Gandhi, Y. Zhang, and C. Fan, "NL2TL: Transforming Natural Languages to Temporal Logics using Large Language Models," in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, H. Bouamor, J. Pino, and K. Bali, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 15 880–15 903. [Online]. Available: https://aclanthology.org/2023.emnlp-main.985/

[24] M. Y. Vardi, "An automata-theoretic approach to linear temporal logic," in *Logics for Concurrency: Structure versus Automata*, F. Moller and G. Birtwistle, Eds. Berlin, Heidelberg: Springer, 1996, pp. 238–266. [Online]. Available: https://doi.org/10.1007/3-540-60915-6_6

[25] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-Logic-Based Reactive Mission and Motion Planning," *IEEE Transactions on Robotics*, vol. 25, no. 6, pp. 1370–1381, Dec. 2009, conference Name: IEEE Transactions on Robotics. [Online]. Available: https://ieeexplore.ieee.org/document/5238617

[26] M. Colledanchise, R. M. Murray, and P. Ögren, "Synthesis of correct-by-construction behavior trees," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2017, pp. 6039–6046, iSSN: 2153-0866. [Online]. Available: https://ieeexplore.ieee.org/document/8206502

[27] R. S. Sutton, D. Precup, and S. Singh, "Between MDPs and semi-MDPs: A framework for temporal abstraction in reinforcement learning," *Artificial Intelligence*, vol. 112, no. 1, pp. 181–211, Aug. 1999. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0004370299000521

[28] N. Yokoyama, A. Clegg, J. Truong, E. Undersander, T.-Y. Yang, S. Arnaud, S. Ha, D. Batra, and A. Rai, "ASC: Adaptive Skill Coordination for Robotic Mobile Manipulation," *IEEE Robotics and Automation Letters*, vol. 9, no. 1, pp. 779–786, Jan. 2024, conference

Name: IEEE Robotics and Automation Letters. [Online]. Available: https://ieeexplore.ieee.org/document/10328058

[29] K. Jatavallabhula, A. Kuwajerwala, Q. Gu, M. Omama, G. Iyer, S. Saryazdi, T. Chen, A. Maalouf, S. Li, N. Keetha, A. Tewari, J. Tenenbaum, C. Melo, M. Krishna, L. Paull, F. Shkurti, and A. Torralba, "ConceptFusion: Open-set multimodal 3D mapping," in *Robotics: Science and Systems XIX*. Robotics: Science and Systems Foundation, July 2023. [Online]. Available: http://www.roboticsproceedings.org/rss19/p066.pdf

[30] Z. Yang, S. S. Raman, A. Shah, and S. Tellex, "Plug in the Safety Chip: Enforcing Constraints for LLM-driven Robot Agents," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*, May 2024, pp. 14 435–14 442. [Online]. Available: https://ieeexplore.ieee.org/document/10611447

[31] I. E. Commission *et al.*, "Functional safety of electrical/electronic/programmable electronic safety related systems," *IEC 61508*, 2000.

[32] M. Minderer, A. Gritsenko, A. Stone, M. Neumann, D. Weissenborn, A. Dosovitskiy, A. Mahendran, A. Arnab, M. Dehghani, Z. Shen, X. Wang, X. Zhai, T. Kipf, and N. Houlsby, "Simple Open-Vocabulary Object Detection," in *Computer Vision – ECCV 2022*, S. Avidan, G. Brostow, M. Cissé, G. M. Farinella, and T. Hassner, Eds. Cham: Springer Nature Switzerland, 2022, pp. 728–755.

[33] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollár, and R. Girshick, "Segment Anything," in *2023 IEEE/CVF International Conference on Computer Vision (ICCV)*, Oct. 2023, pp. 3992–4003, iSSN: 2380-7504. [Online]. Available: https://ieeexplore.ieee.org/document/10378323

[34] A. Majid, M. Bowerman, S. Kita, D. B. M. Haun, and S. C. Levinson, "Can language restructure cognition? The case for space," *Trends in Cognitive Sciences*, vol. 8, no. 3, pp. 108–114, Mar. 2004. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1364661304000208

[35] L. Janson, E. Schmerling, A. Clark, and M. Pavone, "Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions," *The International Journal of Robotics Research*, vol. 34, no. 7, pp. 883–921, June 2015, publisher: SAGE Publications Ltd STM. [Online]. Available: https://doi.org/10.1177/0278364915577958