Automatically Generating Semantic Safety Representations for Indoor Navigation

Haochen Zhang
Robotics Institute
Carnegie Mellon University
Pittsburgh, USA
haochen4@andrew.cmu.edu

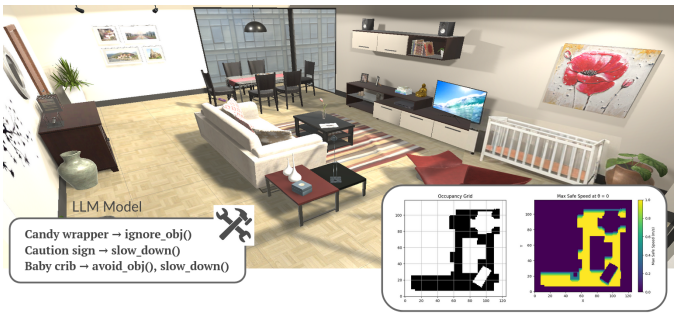


Fig. 1: Overall goal of the project

***Abstract*—For robots operating in human-centered environments such as homes, it is important to be able to autonomously detect and infer commonsense safety constraints that are often trivial to humans. In the indoor navigation domain for example, more nuanced reasoning such as knowing what objects should be avoided and what objects are safe to drive over or what a caution sign entails is desired, but difficult to rigorously express mathematically. Previous works in the domain of robotics navigation have made simplifying assumptions to handle this task, by simplifying the concept of safety and removing semantic nuance, or by limiting the type of safety constraints considered such as purely obstacle avoidance. To tackle this challenge, we propose to leverage the open-world reasoning capabilities of vision language models (VLMs) and large language models (LLMs) to zero-shot infer object-centric safety constraints in indoor scenes. We leverage the commonsense reasoning capabilities of LLMs in the text space along with leveraging tool use to automatically provide an online update of the failure set. With the updated representations, we can then use Hamilton-Jacobi reachability to calculate a backwards reachable tube (BRT) for navigation and obtain a safety controller for any nominal path planner. In this way, the designed approach is both interpretable and rigorous under a safety filtering framework. The approach is tested on a select number of indoor environments and initial results demonstrate the promise of the method for non-trivial constraint reasoning scenarios.**

I. Introduction

As we move towards more widespread deployment of robots in environments with humans, ensuring that they have basic awareness of safety hazards and constraints is important. For robotics applications such as personal assistants or household robots, it is important to be able to navigate in cluttered indoor environments such as homes while handling semantic safety uncertainties or constraints. Moreover, homes are both human-centered and deeply personal environments. Many safety constraints or hazards can be present in home environments that are more complex and nuanced, often requiring some commonsense reasoning which ideally the robot can autonomously detect without being told. For example, slowing down when seeing a wet floor sign (Fig. 2) or moving slower near fragile objects, etc. Cases where the robot is being overly conservative in a dense environment should also be considered (e.g. avoiding a candy wrapper on the ground). Additionally, humans should be able to specify other constraints or preferences using high-level language, such as "Please do not go near the baby's area". While these cases seem trivial to humans, being able to 1) reason about and detect possible constraints or unsafe situations autonomously, and 2) being able to take in human preferences or high-level constraints expressed with natural language online are necessary features for an indoor robot that are not so trivial. Specifically, the task of generating safety representations is more nuanced as it requires a way to map from higher-level semantic concepts of safety in language to a structured representation that can be used with safe control frameworks which ensure safety guarantees.

Fig. 2: A caution sign in a home environment indicating the floor is wet and likely slippery

Prior works aiming to handle this mapping make a number of simplifying assumptions about the definition of safety, or the format of system inputs and outputs. In the indoor navigation domain, previous approaches for safe navigation have traditionally focused on obstacle avoidance or challenging terrain [1], and robust mapping [2], failing to account for

semantic or commonsense safety concepts. Other works that deal with semantic safety [3], [4] simplify the notion of what is considered a failure or constrain the user inputs to the system to be more exact.

In the proposed approach (Figure 1), we leverage the emergent reasoning capabilities of vision language models (VLMs) and large language models (LLMs) to reason about and identify possible hazards in an indoor scene and dictate online updates to a failure set representation by using tool functions. As VLMs have been shown to demonstrate strong capabilities of mapping 2D visual observations into language space [5], we use this to map descriptions of objects in the scene into the text space, where an LLM can reason about the safety constraints and make high-level decisions about how the failure set should be updated. Our key idea is that **incorporating tools with LLMs allow for bridging the gap between semantic understanding to a structured representation of safety constraints in a flexible but precise manner**. While LLMs have been shown to excel at high-level reasoning and planning in the text domain, they have also been shown to struggle with mathematical or numerical reasoning [6]. Thus, the proposed approach makes use of hand-crafted tool functions for modifying the failure set representation at a low-level, which the LLM can call as part of a generated plan. Hamilton-Jacobi reachability analysis [7] is then used to compute the value function as the implicit surface function of the failure set and update a backwards reachable tube (BRT) for the scene online. Using this representation, a downstream policy can then be filtered with this safety controller during path planning to ensure safety. The entire system pipeline is designed to operate online. We evaluate the proposed method on a few example scenes qualitatively by visualizing the BRT, and quantitatively compared to ground-truth representations, showing promising results for using tool functions to convert high-level plans into structured representations in a zero-shot manner.

II. Related Work

A. LLM Tool Use

Our approach is fundamentally inspired by the concept of tool use and task planning in LLMs. One primary work in which this was proposed was Code as Policies [8], which leverages LLMs as high-level task planners to interpret high-level instructions and translate them to low-level code. Other works have also found success with using LLMs to translate general language instructions into structured code or symbolic functions that carry out mathematical calculations to ground a referred object [9], [10]. Closer to our context of reasoning about safety constraints, [11] and [12] use LLMs to parse natural language and translate into logic statements or constraint checks, however, they assume that the safety constraints are explicitly specified in the language instruction without the need for commonsense reasoning.

B. LLMs/VLMs for Online Safety Guidance

Leveraging the powerful reasoning capabilities of foundation models such as LLMs and VLMs, a number of recent works have utilized these models in-the-loop for safety reasoning or correction. FOREWARN [13] uses a VLM online to reason about narrations of action rollouts and choose the safest candidate plan, while [3] uses a VLM to map language constraints into the visual space and use this to then calculate safe states. Most similar to the proposed usage of LLMs/VLMs, AESOP [4] uses LLMs to analyze whether detected objects are safe or anomalous and choose the best control decision. Different from AESOP, rather than just categorizing objects as safe or anomalous based on what was seen in the past, we propose to have the LLM directly reason over what impact a given object should have and generate calls to tool functions to directly update a lower-level safety representation.

C. Safety-Aware Navigation

Prior work in safety-aware navigation in indoor scenes has focused on ensuring socially acceptable navigation around humans [14], navigating complex physical terrain such as stairs [1], and enhancing perception [2] while maintaining robust obstacle avoidance. However, there is limited work into nuanced constraints or "commonsense" object-centric safety constraints that are difficult to mathematically express. Robust obstacle avoidance can also be overconservative, assuming that every detected object must be avoided and thus alter the planned path, which isn't necessarily true.

D. Safety Filtering

The generation of a failure set which can then be used to compute a value function and safety controller for downstream navigation tasks is inspired by the concept of safety filtering, where a baseline or nominal plan is followed until a safety constraint is about to be violated, then the safety filter intervenes with a safety-preserving action. Such safety filters can be computed using a variety of methods such as control barrier functions (CBFs) [15]–[17], or Hamilton-Jacobi (HJ) reachability analysis [3], [7], [18]. In our approach, we use HJ reachability as a rigorous framework to take the updated failure set representation and compute a backwards reachable tube, which can be visualized and used to obtain a safety controller. We follow prior work which deals with computing HJ reachability for policy-agnostic safe control and updating it online through techniques such as warm starting [19], [20].

III. Problem Formulation

Robot System. The robot is a ground vehicle modeled as a 4D Dubin's car with the following state parameters:

$$\dot{x} = v\cos(\theta), \quad \dot{y} = v\sin(\theta), \quad \dot{\theta} = u, \quad \dot{v} = a$$

where (x, y) is the robot's position, θ is its heading, v is the linear velocity, u is the angular velocity, and a is the linear acceleration. The dynamical system at any given time t can then be expressed as:

$$\dot{s(t)} = f(x, y, \theta, v)$$

where $t \in R$, $s \in S$ for all states. We also assume that we have the maximum dimensions of the robot length, width, and height (l, w, h) and that it typically moves with a nominal linear velocity v_{nom}. Similar to [3], we assume the robot may be in an environment with other agents but that they don't interact.

Environment Inputs. The environment consists of M total objects, $O = \{o_1, ...o_M\}$ where the object size, location, and bounding box are known for each object from a mapping module. We assume the robot has seen and mapped the environment once prior to any online constraint updates, which is reasonable for home robots or robots that stay long-term in an indoor environment. The environment is assumed to be an indoor space with negligible external disturbances (e.g. wind, dust, etc.).

We also have input RGB-D images I for the scene from the robot camera, which include image crops of each detected object $\{i_1, ...i_M\} \in I$ as well as input views of the scene from different regions.

A language instruction L is given as input, which can range from being very general (e.g. "Be aware of safety constraints") for default safety awareness to more specific preferences on constraints (e.g. "Be careful near the baby's play area"). This relaxes the assumptions made by previous work [3] where the language input is constrained to be an object name to avoid, which requires human users to specify each individual object to avoid.

Safety Representation. To represent safety constraints for navigation in human-centric environments, we use a failure set F representation for both obstacles and region-dependent speed constraints. Similar to previous work in the navigation domain [2], this includes the scene geometry and traversable area, but extends it to include additional semantic constraints and preference-based failures. Different from previous work in the semantic safety domain [3], the assumption that semantic safety constraints are precisely objects to avoid is relaxed by allowing failures to include regions, speed, and defined by high-level specifications of preference. Furthermore, this failure set is designed to be updated online, which differs from other previous work where the safe set is assumed to be specified ahead of time and unchanged during action execution.

The goal is to obtain a mapping G updatable online that maps from the inputs in the environment to the tightest failure set F:

$$G(I, O, L) = F$$

Tightness means the failure set encodes commonsense safety constraints and constraints important to the user specified through language. The failure set can then be used to compute a BRT and a safety filter for a nominal downstream planner.

IV. Method

The key idea of the proposed approach is that using tools with LLMs enables commonsense safety awareness and adaptable constraint reasoning when generating safety representations. This is based on the idea that LLMs are an excellent interface for taking in natural language instructions and excel at higher-level task planning and reasoning, but struggle with lower-level computation problems that require numerical reasoning or math. Thus, the strength of LLMs as task planners and code generators along with the adaptability of tool functions that handle the mathematical reasoning can be jointly leveraged with tool use.

The overall method consists of the following modules: 3D perception and mapping, 2D perception, LLM reasoning and representation generation, and online constraint updating. Each component is further explained below along with their design decisions and an overall system diagram is shown in Figure 3.

A. 3D Perception and Mapping

An off-the-shelf 3D mapping and perception module [21] is used to gather information about the 3D scene. The module is an open-vocabulary semantic mapping module that uses 360 panoramic images, 3D LiDAR scans, and odometry as inputs. As the robot explores a scene for the first time, the Grounded-SAM-2 [22] model and real-time tracking module are used to take in the sensor inputs and runs object detection and segmentation, generating an object-level map of the scene that consists of the scene point cloud, object bounding boxes, and labels. A map of the free traversable space in the scene is also obtained after exploring the environment by projecting detected object bounding boxes on the ground to the x-y plane. As the application is intended for indoor environments such as homes, we make the assumption that the robot will have explored the environment previously at least once. The objects in the 3D scene are then stored in a scene graph representation and trivial objects are filtered out, such as the ceiling, floor, etc.

B. 2D Perception and Semantics

In parallel with the 3D mapping during an initial exploration of the scene, we implement a 2D VLM-based captioning module that runs online simultaneously. As objects are detected by the semantic mapper and added to the map, 2D RGB images of the object are captured and filtered using CLIP [5] to find the image crop containing the "best view" of the image, represented as the highest CLIP embedding similarity to it's text label. The object image is then passed into a local Qwen2 [23] model with a template prompt to generate a descriptive visual caption, which we assume is enough to reason about the semantic safety implications of the object. The visual captions are stored for each object in the scene graph. We use the Qwen2-7b model because it can be run locally without being computationally expensive and can be prompted to generate captions adhering to a specific format. The object captioning stage is designed to be decoupled from the LLM reasoning stage to allow for different frequency of calls to these foundation models. Pre-generating these captions and storing them as objects are mapped maximizes efficiency when running the system online.

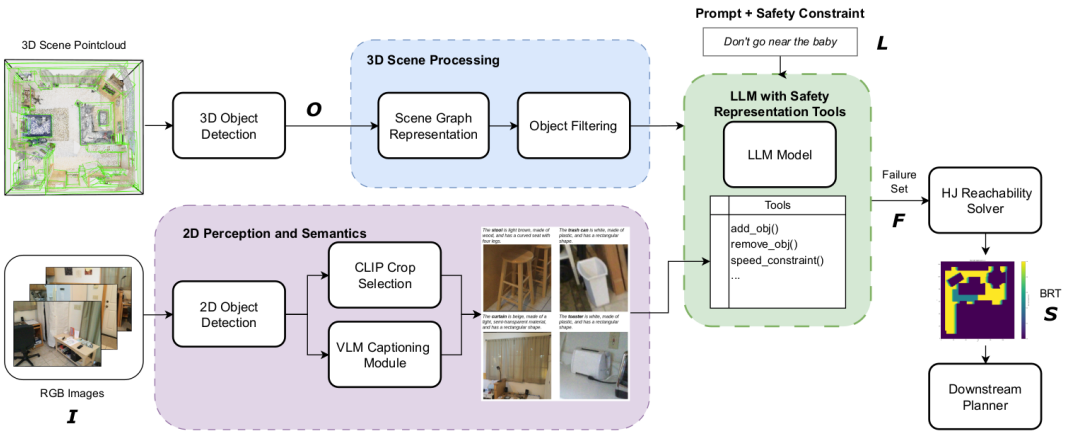


Fig. 3: Full pipeline of proposed method with 2D perception, 3D mapping, and LLM with tools reasoning

C. LLM Reasoning with Tools

We use an LLM to take in natural language instructions, reason about safety concepts requiring commonsense reasoning, dictate how to update a failure set, then make the function calls to do so. To do this, we abstract away the representation updating by providing a set of tool functions that handle the computation and provide them as input to the LLM. These tool functions are handcrafted to be a *fundamental set of elementary functions that update the failure set* in different ways by applying them on a specific object or region to update the failure $l(x)$ function. Specifically, we consider navigation-related failures such as collision with obstacles, entering restricted areas, and surpassing speed constraints. On the other hand, to avoid overly conservative behavior, we also allow for the removal of objects from the failure set that are trivial to consider for safe navigation.

The tool functions and their descriptions are shown below.

- `add_object()`: adds an object to the failure set to avoid based on an object bounding box
- `remove_object()`: removes an object from the failure set that can be ignored based on an object bounding box
- `add_area()`: adds a region of space to the failure set to avoid based on an area polygon
- `remove_area()`: removes a region of space from the failure set that can be ignored based on an area polygon
- `slow_down()`: adds a maximum speed constraint to a certain region of space

The LLM takes as input 1) a language-specification of a safety constraint, 2) object information from the perception modules in text format, and 3) a system prompt. In the system prompt, it is given the set of tool functions that update the failure $l(x)$ along with function docstrings describing they input and output, and some few-shot examples. The LLM is then instructed to generate code that calls one or more tool functions to update the safety representation for each object given. We use the Mistral Large model [24] as the LLM and assume the robot agent has internet connection, which is a reasonable assumption to make for indoor, human-centric environments.

D. BRT Generation

With the updated failure set represented as the level set of some $l(x)$ as the safety representation updated by the LLM function calls, the BRT is generated using HJ reachability based on solving the Hamilton-Jacobi-Isaacs Variational Inequality (HJI-VI) to compute the target value function $V(s,t)$:

$$\min\left\{\frac{\partial V}{\partial t}(s,t) + H(\tau, s, \nabla V(s,t)), g(s) - V(s,t)\right\} = 0,$$

$$V(s,0) = g(s), t \leq 0$$

The inequality is solved using a numerical PDE solver. Using the value function $V(s,t)$ solved from the above inequality, the BRT can be computed as the sub-zero level set of V:

$$S_{unsafe}(t) = \{s : V(s,t) \leq 0\}$$

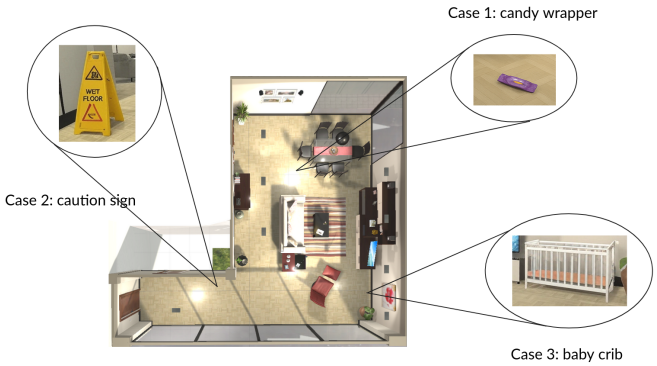
The safe states are thus $S_{safe} = (S_{unsafe})^{\complement}$. The BRT is computed online using warm-starting techniques where the value function computed previously is used as a starting point for the current BRT computation to maximize efficiency.

With this BRT, a minimally invasive safety feedback controller $\pi_{safe}(s)$ can also obtained, which can then be used to filter a nominal path planner in a downstream task.

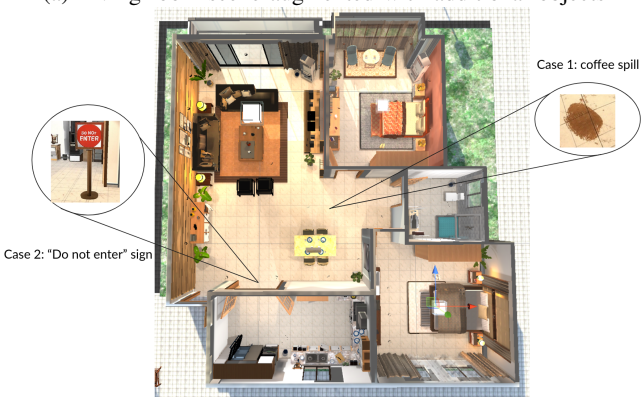
V. Experimental Setup

To evaluate the proposed method, a few selected indoor 3D scenes custom rendered in the Unity game engine [25] are augmented with some additional objects that lead to semantic safety constraints to best evaluate the method on commonsense safety reasoning scenarios that might not exist in the scenes by default. One sample living room scene was augmented with a wet floor sign, a candy wrapper, and a baby crib as shown in Figure 4a. Another larger scene for a home with multiple rooms was augmented with a coffee spill and a "Do Not Enter" sign near the kitchen as shown in Figure 4b. We assume the

robot has explored the scene already and has a prior floor map of the scene to use as warm-starting, and only provide the newly added objects to the LLM. The method, however, is designed such that the LLM could reason about each detected object in the scene as it's detected.



(a) Living room scene augmented with additional objects



(b) Home building scene augmented with additional objects

Fig. 4: Overhead view of test scenes with added objects

The method is first evaluated qualitatively by visualizing the generated BRT. Quantitatively, the similarity between the automatically generated target value function and the ground-truth target value function will be compared with a similarity score. The ground-truth value function is manually annotated for the scenes and a similarity score of 1.0 indicates an exact match with the generated values. The average time it takes the LLM to reason about the safety constraints and update the failure set, as well as generation time for the BRT will also be measured. Finally, we ablate each component of the proposed pipeline on one of the scenes and demonstrate the results both qualitatively and quantitatively. In general, the quality of such semantic safety representations in personal environments would benefit from human evaluation as the safety constraints need to align with human notions of safety and personal preferences. After integrating with a downstream path planner, downstream navigation metrics such as cost to goal, navigation success, and plan time can also be used a metrics for evaluation.

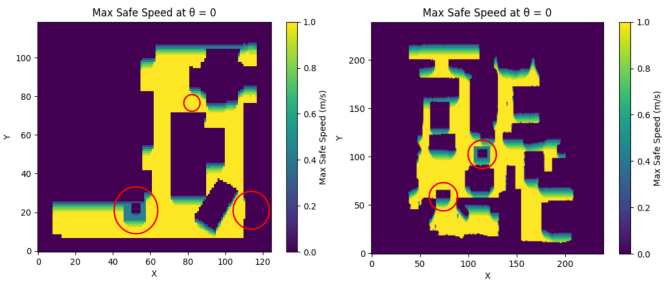
VI. Results

A. Qualitative Results

The original occupancy grid based on the floormap and existing objects is shown in Figure 5. The final generated BRT for the two scenes are visualized in Figure 6 with circles highlighting the objects that the LLM reasoned about and updated the BRT correspondingly. The color bar on the right indicates speed constraints by showing the maximum safe speed allowed, where 1.0 (shown as yellow) is the nominal speed. The prompt given to the LLM is the default prompt to reason about safety constraints for the newly added objects.

(a) (b)

Fig. 5: Occupancy grid based on floorplan for the a) living room scene and b) the home scene



(a) (b)

Fig. 6: Generated BRT for the a) living room scene and b) the home scene with automatic updates to the BRT in red circles

For the living scene, the LLM reasoning module added the wet floor sign as an obstacle and enforced a slower speed around the object, added the baby crib as an obstacle, and chose to ignore the candy wrapper and updated the failure set by successfully calling the corresponding tool functions for each object. Overall, the method is shown to have common-sense safety awareness such as slowing down near the wet floor sign and ignoring objects trivial to navigation such as a wrapper on the ground. One thing to note is that while the LLM was not enforced to only call one tool function for each object, it often does so and did not enforce any speed constraints near the baby crib, which could be a semantic safety constraint for a human user. Additionally, speed constraints

near the caution sign are only enforced around the object bounding box and lacks the regional awareness that humans have about where it may be in effect.

For the home building scene, the reasoning module updated the failure set by adding both objects as objects to avoid. However, this is farther from the semantic reasoning that humans would be capable of. Particularly, when seeing a "Do Not Enter" sign (position indicated by the lower red circle), humans know that the sign applies to the entire area behind it when facing the sign. However, such viewpoint grounding and spatial awareness is a limitation of the current LLM and tools configuration, where the LLM does not have the tools or spatial knowledge to update the value function by blocking off all traversable area past the sign and only enforced the area the sign occupies as an obstacle.

With the current pipeline, missing constraints or unideal behaviors can be easily added as a high-level language instruction onto the prompt, such as "be careful near the baby" for the living room scene, which results in the speed constraint being added to the region near the baby crib as shown below on the right of Figure 7. High-level language constraints are favored as they offer more usability and practicality to human users compared to requiring humans specify objects to individually avoid.

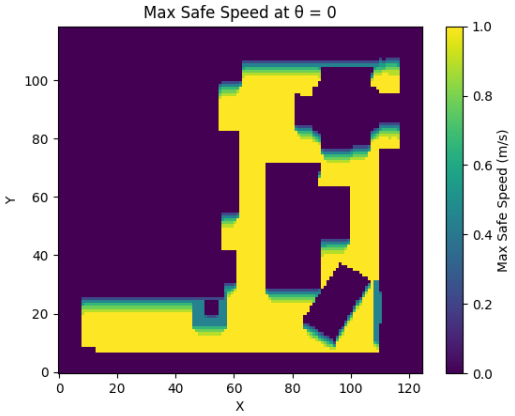


Fig. 7: Generated BRT for the living room scene with an additional language instruction

B. Quantitative Results

Initial quantitative results on the two scenes are shown in Table I where average scores and standard deviation are obtained from three trials. We evaluate the case where the LLM is only given a default instruction to reason about safety constraints without specifying particular constraints. The similarity score does not have standard deviation numbers as the LLM called the same tool functions each time for the limited number of objects tested, resulting in the same $l(x)$ updates. For both scenes, the similarity score is relatively high compared to ground-truth, likely due to the small number of objects evaluated compared to the size of the scene. For the home building scene in particular, the score is lower due to not enforcing the entire kitchen area behind the "Do Not Enter" sign as an obstacle.

The LLM reasoning and tool calling time for both scenes is relatively low, likely due to few objects being evaluated and relatively small grid sizes for both scenes as this time is directly correlated to how many constraints are being updated. This provides a reasonable estimate for the inference time needed during deployment, as we expect this reasoning module to be triggered when new objects are detected, only requiring it to handle a few objects at once. However, this time is expected to scale up for larger scenes and if more tool functions are added.

The BRT generation time is highly dependent on the size of the scene as evidenced by the much larger computation time for the larger home building scene and stands as a computation limitation for the system, even with using warm-starting techniques by using the pre-computed value function for the scene from the prior timestep without the newly added objects.

Scene	**Similarity Score**	**LLM Reasoning Time (s)**	**BRT Generation Time (s)**
Living room	0.98	2.23 $\pm$0.12	5.07 $\pm$2.57
Home building	0.92	2.01 $\pm$0.16	13.71 $\pm$3.89

TABLE I: Quantitative results on two scenes

C. Ablation Studies

We further ablate each modular component of the proposed pipeline, namely the VLM-generated captions, the tools provided to the LLM, and the additional language instruction on the living room scene. To ablate the additional language instruction, the prompt is left to be generic "What safety constraint(s) if any should be used?". To ablate the visual captions given to the LLM, only the object class name from the mapping module is passed into the LLM. To ablate the tool functions, The LLM is used as an end-to-end planner, given the bounding box coordinates of each object in the prompt, an explanation of the failure set array, and asked to generate a failure set directly.

For each ablation test, we run the computation three times and report the average similarity score of the generated value function to a ground-truth value function that's manually annotated, as well as the average time it takes the LLM module to reason about the constraints and generate the output failure set. The quantitative results are shown in Table II.

Tool Functions	Visual Captions	Language Instruction	**Similarity Score**	**LLM Reasoning Time (s)**
			0.954	17.4 $\pm$5.72
✓			0.970	2.01 $\pm$0.16
✓	✓		0.981	2.23 $\pm$0.12
✓	✓	✓	**0.984**	1.98 $\pm$0.16

TABLE II: Ablation studies on the living room scene

From the quantitative results, we observe that the full pipeline with all components has the highest similarity score to

the ground-truth human-annotated result, indicating the necessity of tool use and visual captions, and benefit of high-level preference instructions. The biggest jump between similarity scores is with the inclusion of visual captions, which follows the intuition that visual information is critically important for making semantic safety judgments. For example, without visual captions, the caution sign is only labeled by it's class name from the mapper, and passed into the LLM as "sign", losing the semantic meaning related to the sign. With visual captions, the VLM-generated caption passed into the LLM is "The sign is yellow, plastic material, and triangular in shape. It has a bright, reflective surface with bold black text that reads WET FLOOR and features warning symbols, indicating to proceed carefully", which already has constraint/affordance reasoning in the much richer text description. It is important to note that the similarity scores are still relatively close to each other, likely due to the fact that the scores are only based on three added objects in a relatively small scene.

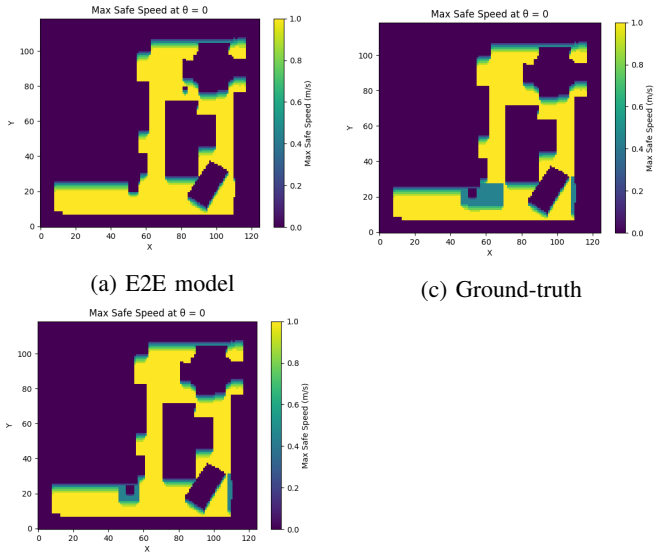
In terms of LLM reasoning time, the time taken is significantly longer for the end-to-end model that does not use tools. Qualitatively, it was observed to require longer to generate a failure representation directly and required multiple steps in chain-of-thought prompting to arrive at a result.

Qualitatively, the generated BRT for the end-to-end LLM with all components ablated (first row of the table) is shown in Figure 8a while the generated BRT for the pipeline with all components (last row of the table) is shown in Figure 8b. For a visual comparison, the ground-truth BRT is shown in Figure 8c. With the end-to-end model, the LLM simply adds all objects as an obstacle to the BRT, which is less semantically nuanced than a human user would ideally want as there is no speed change around the wet floor sign and the candy wrapper is added as an obstacle. Comparing the full model's performance, the generated BRT is close to the ground-truth except the speed constraint is only applied near the wet floor sign while humans usually know to take caution in the region following the sign.

VII. Limitations and Future Work

Despite the promising results shown on the two sample scenes, a number of limitations exist with the current method. As with any method that uses pre-trained foundation models such as VLMs and LLMs, the performance is limited by the performance of these models and highly sensitive to the prompt given, especially the LLM reasoning module that calls tool functions. Future work could relate to how best to prompt-tune these models or even fine-tune these models to specific domains such as navigation.

Another limitation with the current method is the use of handcrafted tools, as they limit the ways in which the failure set can be updated and require a user to predefine these. There are also tradeoffs between having lots of highly specific functions which can help with more nuanced ways of updating the failure set, the LLM's ability to distinguish and call these functions, and the generalizability to other domains outside of navigation.



(a) E2E model

(c) Ground-truth

(b) Full pipeline

Fig. 8: Generated BRT for different ablation models vs. ground-truth

Finally, the proposed method demonstrates the initial feasibility of using tools with LLMs to bridge the gap between semantic reasoning and structured safety representations, but could benefit from larger scale evaluation on more diverse scenes and scenarios, and use humans to score the quality of the representation. In addition, integrating the pipeline with a downstream path planner in real-time and evaluating the safety filtering with the navigation task will help validate the representations generated. Further deploying the pipeline on a physical system would be paramount to evaluating the performance online and how components assumed to be perfect, such as the perception, can affect the safety representations generated. In the interim, the generated representations have been used to provide an updated floor plan to a nominal path planner as an initial visualization of the planned future work (Figure 9).

Fig. 9: Planned path for the living room scene with the vehicle avoiding the caution sign and ignoring the wrapper

VIII. Conclusion

For robot navigation agents in personal home environments, the ability to both commonsense reason about semantic safety constraints and take in high-level instructions for constraints are important, but connecting these to lower-level safety representations is difficult. In this work, we propose a pipeline that bridges this gap in a zero-shot manner by leveraging an LLM equipped with a set of lower-level tool functions to reason about object-centric constraints with pre-trained open-world knowledge and use the tool functions to automatically update a failure set. The failure set can then be used to compute a BRT for safety filtering any nominal path-planning policy. The promising results of this approach for the indoor navigation domain are shown qualitatively and quantitatively on two scenes with a select number of object case studies.

References

[1] C. Wang, J. Wang, C. Li, D. Ho, J. Cheng, T. Yan, L. Meng, and M. Q.-H. Meng, "Safe and robust mobile robot navigation in uneven indoor environments," *Sensors*, vol. 19, no. 13, p. 2993, 2019.

[2] A. C. Victorino, P. Rives, and J.-J. Borrelly, "Safe navigation for indoor mobile robots. part i: a sensor-based navigation framework," *The International Journal of Robotics Research*, vol. 22, no. 12, pp. 1005–1118, 2003.

[3] L. Santos, Z. Li, L. Peters, S. Bansal, and A. Bajcsy, "Updating robot safety representations online from natural language feedback," *arXiv preprint arXiv:2409.14580*, 2024.

[4] R. Sinha, A. Elhafsi, C. Agia, M. Foutter, E. Schmerling, and M. Pavone, "Real-time anomaly detection and reactive planning with large language models," *arXiv preprint arXiv:2407.08735*, 2024.

[5] 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*. PmLR, 2021, pp. 8748–8763.

[6] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[7] S. Bansal, M. Chen, S. Herbert, and C. J. Tomlin, "Hamilton-jacobi reachability: A brief overview and recent advances," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2242–2253.

[8] 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)*. IEEE, 2023, pp. 9493–9500.

[9] J. Hsu, J. Mao, and J. Wu, "Ns3d: Neuro-symbolic grounding of 3d objects and relations," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 2614–2623.

[10] J. Hsu, J. Mao, J. Tenenbaum, and J. Wu, "What's left? concept grounding with logic-enhanced foundation models," *Advances in Neural Information Processing Systems*, vol. 36, pp. 38 798–38 814, 2023.

[11] 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)*. IEEE, 2024, pp. 14 435–14 442.

[12] F. Lotfi, F. Faraji, N. Kakodkar, T. Manderson, D. Meger, and G. Dudek, "Constrained robotic navigation on preferred terrains using llms and speech instruction: Exploiting the power of adverbs," in *International Symposium on Experimental Robotics*. Springer, 2023, pp. 118–128.

[13] Y. Wu, R. Tian, G. Swamy, and A. Bajcsy, "From foresight to forethought: Vlm-in-the-loop policy steering via latent alignment," *arXiv preprint arXiv:2502.01828*, 2025.

[14] D. Song, J. Liang, A. Payandeh, A. H. Raj, X. Xiao, and D. Manocha, "Vlm-social-nav: Socially aware robot navigation through scoring using vision-language models," *IEEE Robotics and Automation Letters*, vol. 10, no. 1, pp. 508–515, 2025.

[15] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *2019 18th European control conference (ECC)*. Ieee, 2019, pp. 3420–3431.

[16] Y. Chen, A. Singletary, and A. D. Ames, "Guaranteed obstacle avoidance for multi-robot operations with limited actuation: A control barrier function approach," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 127–132, 2020.

[17] J. Li, Q. Liu, W. Jin, J. Qin, and S. Hirche, "Robust safe learning and control in an unknown environment: An uncertainty-separated control barrier function approach," *IEEE Robotics and Automation Letters*, vol. 8, no. 10, pp. 6539–6546, 2023.

[18] S. Singh, A. Majumdar, J.-J. Slotine, and M. Pavone, "Robust online motion planning via contraction theory and convex optimization," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 5883–5890.

[19] A. Bajcsy, S. Bansal, E. Bronstein, V. Tolani, and C. J. Tomlin, "An efficient reachability-based framework for provably safe autonomous navigation in unknown environments," in *2019 IEEE 58th Conference on Decision and Control (CDC)*. IEEE, 2019, pp. 1758–1765.

[20] J. Borquez, K. Nakamura, and S. Bansal, "Parameter-conditioned reachable sets for updating safety assurances online," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 10 553–10 559.

[21] G. Chen, C. Yao, W. Wang, and J. Zhang, "Semantic mapping with 360 camera and 3d lidar," https://github.com/gfchen01, 2025.

[22] T. Ren, S. Liu, A. Zeng, J. Lin, K. Li, H. Cao, J. Chen, X. Huang, Y. Chen, F. Yan *et al.*, "Grounded sam: Assembling open-world models for diverse visual tasks," *arXiv preprint arXiv:2401.14159*, 2024.

[23] Q. Team, "Qwen2 technical report," 2024. [Online]. Available: https://arxiv.org/abs/2407.10671

[24] M. A. Team, "Mistral large 2: The new generation of flag- ship model," https://mistral.ai/news/mistral-large-2407/, 2024, [Accessed 01-03-2025].

[25] Unity Technologies, "Unity," 2023, game development platform. [Online]. Available: https://unity.com/