# From Words to Safety: Language-Conditioned Safety Filtering for Robot Navigation

Zeyuan Feng[1], Haimingyue Zhang[2], Somil Bansal[1]

***Abstract*— As robots are increasingly deployed in open-world, human-centered environments, their ability to interpret natural language instructions and adhere to safety constraints is critical for effective and trustworthy interaction. Existing approaches often focus on mapping language to reward functions, rather than explicit safety specifications, or they address narrow constraint classes (e.g., obstacle avoidance), limiting their robustness and generality. We propose a modular framework for language-conditioned safety in robot navigation that enables free-form language instructions to define enforceable safety constraints. Our framework is composed of three core components: (1) a large language model (LLM)-based semantic parser that translates free-form instructions into structured safety specifications; (2) a perception module that grounds these specifications through object-level 3D scene representations; and (3) a model predictive control (MPC)-based safety filter that enforces both semantic and geometric constraints in real time, independent of the underlying task policy. We evaluate the effectiveness of the proposed framework through both simulation studies and hardware experiments, demonstrating that it robustly interprets and enforces diverse language-specified constraints across a wide range of environments, enabling a principled integration of natural language and safety-critical control.**

## I. Introduction

Robots are increasingly deployed in human-centered environments such as homes, hospitals, and workplaces, where safety is fundamental to trust and widespread adoption. Existing approaches predominantly address static, predefined safety constraints, most notably collision avoidance. Through methods such as Model Predictive Control (MPC) [1], [2], Control Barrier Functions (CBF) [3], [4], and reachability analysis [5], [6], classical techniques have proven highly effective at filtering unsafe actions. However, these approaches presuppose fixed safety specifications, limiting their ability to adapt to evolving contexts and user expectations.

In contrast, failures in open-world deployments are often *dynamic* and *semantic*: they arise from context- and preference-dependent constraints that evolve with the situation and the user. We refer to this broader requirement as ***semantic safety*** – the ability of robots not only to avoid collisions, but also to anticipate and adapt to context-specific safety expectations expressed by humans around them. For example, a nurse may instruct a medical delivery robot to "*be quiet around the patient's bed*" when a patient should not be disturbed. Such constraints are difficult to anticipate and encode *a priori*, yet are naturally expressed in language by non-technical users. Enabling robots to accept, ground, and enforce such natural language constraints at runtime is therefore a key missing capability.

Recent progress in vision–language models (VLMs) has primarily focused on high-level decision making, such as specifying tasks and goals in natural language or shaping reward functions. In contrast, vision-language-conditioned safety remains underexplored. Prior efforts are limited either to policy steering or to handling a narrow class of language-specified constraints. Consequently, the central challenges of language-based safety remain open, i.e., (1) grounding vague or ambiguous instructions in perception, (2) generalizing across diverse instructions and constraint types, (3) maintaining interpretable and auditable safety specifications, and (4) enforcing them in real time during execution.

To address these challenges, we introduce a modular framework for language-conditioned safety. Our system integrates three components: (1) a language module that parses free-form instructions into structured safety configuration files; (2) a perception module that persistently grounds referenced objects and regions in 3D; and (3) a sampling-based safety filter that enforces novel semantic and geometric constraints online while minimally deviating from nominal control. Together, these components enable robots to incorporate runtime language instructions into their low-level safety guarantees (Fig. 1). We validate our framework extensively in both simulation and hardware experiments, demonstrating robustness across diverse environments and instruction types. Ultimately, our framework leverages VLMs as open-world runtime monitors, with natural language serving as a flexible interface for specifying evolving safety constraints.

## II. Related Work

**Language for Constraints vs. Rewards.** Most language-conditioned robotics methods leverage LLM reasoning for reward design and shaping [7]–[10]. While effective for guiding behavior, these approaches typically treat safety as a soft cost within an optimization objective, rather than as a strict requirement, or omit safety considerations altogether. In contrast, only a limited number of works attempt to derive explicit safety constraints from language. For instance, [11] aligns language with learned world models for safe policy steering, but relies on substantial manual engineering and remains domain-specific. [12] integrates VLMs with Hamilton–Jacobi (HJ) reachability for semantic safety filtering, though it addresses only obstacle-avoidance-style constraints. [13] employs VLMs to infer relevant semantic constraints for manipulation tasks and enforces them via CBF-based filters. Our work complements these efforts by targeting

This research is supported in part by the DARPA Assured Neuro Symbolic Learning and Reasoning (ANSR) program and by the NSF CAREER program (2240163). [1]Authors are with the department of Aeronautics and Astronautics, Stanford University, Stanford, USA. [2]Author is with the School of Vehicle and Mobility at Tsinghua University, Beijing, China, and Corresponding author: {zeyuanf@stanford.edu}

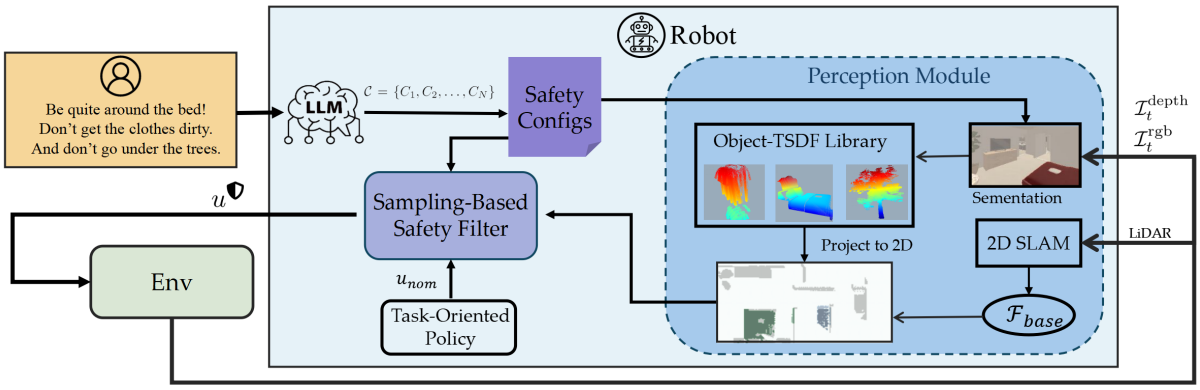



Fig. 1: Framework Overview. Language instructions are parsed by an LLM into structured safety configuration files; the referenced objects/regions are grounded via the perception module, and enforced by a sampling-based safety filter.

semantic safety for navigation in unstructured environments, scaling to diverse constraint types, and enabling runtime template-based specification parsing coupled with online safety filtering.

**Grounding Language in Perception.** Translating instructions such as "*be quiet around the bed*" into actionable safety constraints requires robust perceptual grounding. Prior approaches typically rely on either (i) semantic SLAM or scene-graph construction [14]–[17], or (ii) open-vocabulary segmentation combined with geometric fusion [18]–[21]. The former provides rich global representations but is computationally demanding and sensitive to odometry drift. For real-time navigation, we adopt the latter, while remaining compatible with advances in both paradigms.

**Safety Filtering.** Safety filtering methods modify a robot's nominal control in real time to enforce safety constraints. Control-theoretic approaches such as HJ reachability and CBFs [4], [5], [22] provide strong theoretical foundations for runtime shielding. However, reachability methods scale poorly to dynamically evolving specifications, especially in higher dimensions [12], [23], and practical CBFs often rely on heuristic or learned barriers that are less robust when constraints change online. Since our constraints evolve as new instructions are parsed, we adopt a predictive safety filter inspired by Model Predictive Path Integral (MPPI) control [24]. While it lacks formal guarantees, it empirically enables effective runtime enforcement under evolving semantic constraints.

## III. Problem Setup

We consider a mobile robot navigating in a static environment $\mathcal{E}$ while executing a nominal, goal-directed navigation policy. The robot evolves under discrete-time dynamics $x_{k+1} = f(x_k, u_k)$, where $x \in \mathbb{R}^n$ denotes the state and $u \in \mathbb{R}^m$ is the control input. At each time step $t$, the robot has access to: (i) a nominal control command $u_{\text{nom}}$ from a task planner; (ii) RGB-D observations $(\mathcal{I}_t^{\text{rgb}}, \mathcal{I}_t^{\text{depth}})$; (iii) an estimated state $\hat{x}_t$ from onboard odometry; and (iv) a base failure set and signed distance function $\mathcal{F}_{base} = \{x \mid l_{base}(x) \leq 0\}$, representing geometric obstacles (e.g., detected by a lightweight 2D LiDAR-based SLAM module). The nominal control policy, $u_{\text{nom}}$, is task-oriented and may violate safety constraints.

In addition, the robot may receive natural language instructions at runtime. Specifically, the robot maintains a set of active instructions $\mathcal{L} = \{L_1, L_2, \ldots, L_p\}$, which may be empty initially and evolve over time as new commands are issued. These instructions encode semantic safety constraints $\mathcal{C}_{\text{sem}}$ (e.g., "*move slowly near the crib*"), augmenting the geometric collision constraints $\mathcal{C}_{\text{base}}$.

Our objective is to design a closed-loop, language-aware safety filter, $\pi_{\text{safe}} : (x_t, u_{\text{nom}}, \mathcal{L}) \mapsto u_{\text{filtered}}$, that minimally modifies $u_{\text{nom}}$ to ensure real-time compliance with both $\mathcal{C}_{\text{base}}$ and $\mathcal{C}_{\text{sem}}$. The filter must satisfy three requirements: (a) *interactive operation* — run at control-loop frequencies to ensure responsiveness; (b) *robustness* — tolerate uncertainty in perception and state estimation; and (c) *generality* — handle diverse categories of language-specified constraints.

## IV. A Modular Framework for Handling Natural Language Safety Instructions

**Framework Overview.** The proposed system consists of an LLM-based instruction parser, a 3D perception grounding module, and a predictive safety filter (Fig. 1). The language module runs asynchronously upon new instructions, the perception module at the sensing rate, and the safety filter at every control step.

### A. Safety Specification Synthesis (Language Module)

We categorize language-specified safety constraints into three broad classes that cover the diverse types of instructions encountered in navigation tasks:

***Spatial Exclusion Constraints*** - prohibit the robot from entering, passing beneath, or approaching certain regions or objects (e.g., "*do not go beneath tables*). These unify classic avoidance, overhead/underfoot, and buffer-margin directives.

***Kinematic Modulation Constraints*** — regulate the robot's motion either globally or within specific contexts (e.g., "*slow down on the carpet*").

***Abstract Intent Constraints*** — express high-level safety guidance in vague or subjective terms (e.g., "*be quiet around the bed*"). Such instructions require interpretation and are resolved into spatial exclusion or kinematic modulation primitives through LLM reasoning.

To unify these categories, we design a JSON-style specification template:

```
T = {"type": str, "obj": str,
    "buffer": float, "vel max": float,
    "angular vel max": float}
```

where each field encodes the constraint type, referenced objects, buffer distance, and velocity limits.

Given a natural language instruction $L$, we translate it into a structured configuration $C$ that conforms to this template. The translation is performed by an LLM prompted with the schema definition, a description of the robot's capabilities, and a set of few-shot examples. The robot maintains a dynamic list of active configurations $\mathcal{C} = \{C_1, C_2, \ldots, C_N\}$, capped at $M$ entries to prevent unbounded growth. When instructions are ambiguous, underspecified, or inconsistent with the robot's capabilities, the LLM is instructed to explicitly express uncertainty in its interpretation and request clarification before proceeding. The resulting configurations in $\mathcal{C}$ serve as structured, interpretable inputs for both the perception module and the safety filter. Parsing runs asynchronously from the control loop, so sparse, low-frequency language inputs incur no per-timestep overhead and do not compromise real-time safety enforcement.

### B. Constraint Grounding (Perception Module)

Once the set of safety specifications $\mathcal{C}$ is generated by the language module, the perception module grounds each specification $C_i$ into a semantic failure set $\mathcal{F}^i_{\text{sem}}$ for downstream safety filtering. Our design emphasizes computational and memory efficiency, avoiding the overhead of full 3D semantic SLAM while still enabling persistent, semantically grounded object tracking.

**3D Representation Generation:** We adopt a hybrid segmentation strategy that combines the strengths of open-vocabulary and panoptic segmentation. CLIP-based segmentation [25] provides open-vocabulary recognition but coarse object boundaries, while panoptic segmentation yields accurate object boundaries but lacks semantic label assignment. For each safety specification and RGB frame, we compute both segmentation outputs and fuse them to produce semantic masks $M^i_t = \text{Seg}(\mathcal{I}^{\text{rgb}}_t, C_i)$. Masked RGB-D observations are then projected into a 3D point cloud using camera intrinsics and robot pose estimated by a SLAM module $\text{proj}\left(\mathcal{I}^{\text{depth}}_t, M^i_t, x\right) \rightarrow \mathbf{p}^i_t$ and integrated into a voxel-based truncated signed distance function (TSDF) representation $\mathbf{o}^i$, enabling persistent object tracking associated with $C_i$. Any perception pipeline producing equivalent representations can be used within the proposed framework.

**Grounding Semantic Failure Set:** Finally, each specification $C_i$ is mapped to a semantic failure set $\mathcal{F}^i_{\text{sem}} = \{l_i(x) \leq 0\}$, where $l_i(x)$ is a constraint-specific scalar function. For instance:

$$\begin{aligned}
\text{Spatial Exclusion:} \quad & l_i(x) = \text{SDF}(x; \mathcal{O}_i) - \text{buffer}_i, \\
\text{Kinematic Modulation:} \quad & l_i(x) = \min(v^i_{\max} - v, \omega^i_{\max} - \omega), \\
\text{Hybrid Constraints:} \quad & l_i(x) = \max\left(\text{SDF}(x; \mathcal{O}_i) - \text{buffer}_i,\right. \\
& \left. \min(v^i_{\max} - v, \omega^i_{\max} - \omega)\right),
\end{aligned}$$

where SDF denotes the signed distance function, $\mathcal{O}_i$ is a 2D occupancy map projected from each object-level TSDF, and $\text{buffer}_i$, $v^i_{\max}$, and $\omega^i_{\max}$ denote the constraint parameters associated with $C_i$. The overall semantic failure set is then

$$\mathcal{F}_{\text{sem}} = \cup_{i=1}^{N} \mathcal{F}^i_{\text{sem}} = \{min_i \; l_i(x) \leq 0\}. \tag{1}$$

### C. Safety Filtering

To ensure constraint satisfaction at runtime, we introduce two sampling-based, MPC-inspired safety filters – analogous to least-restrictive and smooth-blending safety filters [5] – that enforce safety while minimally deviating from the nominal control. Compared to existing CBF-based or HJ reachability-based filters, which rely on precomputed value functions or fixed constraint sets, the proposed sampling-based predictive filters offer greater flexibility for handling dynamically generated language instructions and heterogeneous constraint types (e.g., geometric obstacles, velocity bounds, or hybrid combinations). Both filters share a common backbone: a safety score computation and a control filtering policy.

**Safety Score Computation.** Given an initial state $x_{\text{init}}$, the safety score is computed by solving the following finite-horizon optimization problem:

$$\begin{aligned}
V^{\text{safe}} = & \max_{u_{0:H-1}} \min_{k \in \{0,1,\ldots,H\}} l\left(x_k\right) \\
& \text{s.t. } x_{k+1} = f\left(x_k, u_k\right), \; u_k \in \mathcal{U}, \; x_0 = x_{\text{init}},
\end{aligned} \tag{2}$$

where $l(x) := \min(l_{\text{sem}}(x), l_{\text{base}}(x))$ encodes the combined semantic and geometric safety constraints. Its sub-zero level set corresponds to the overall failure set, i.e., $\mathcal{F}_{\text{base}} \cup \mathcal{F}_{\text{sem}} = \{x : l(x) \leq 0\}$. Intuitively, the cost function represents the minimal safety margin over the horizon, which the optimization seeks to maximize. If $V^{\text{safe}} \leq 0$, no safe backup trajectory exists. The optimal safe control at $x_{\text{init}}$ is given by $u^{\text{🛡}} := u^*_0$.

In this work, we approximate this optimization via a sampling-based MPC (SB-MPC) procedure (Alg. 1). At each iteration, $N$ control sequences are sampled around the current best candidate, rolled out over horizon $H$, and scored by their minimum safety margin. The "safest" sequence is then used to warm-start the next iteration. This iterative refinement process guarantees a non-decreasing safety score across $R$ iterations and returns the best trajectory discovered.

**Least-Restrictive Safety Filter.** The least-restrictive filter executes the nominal control whenever its predicted trajectory maintains a positive safety margin above a threshold $\epsilon$. Otherwise, it switches to the backup safe control obtained from SB-MPC in Alg. 1. Specifically, let $V^{\text{safe}}_{\text{nom}}$ denote the

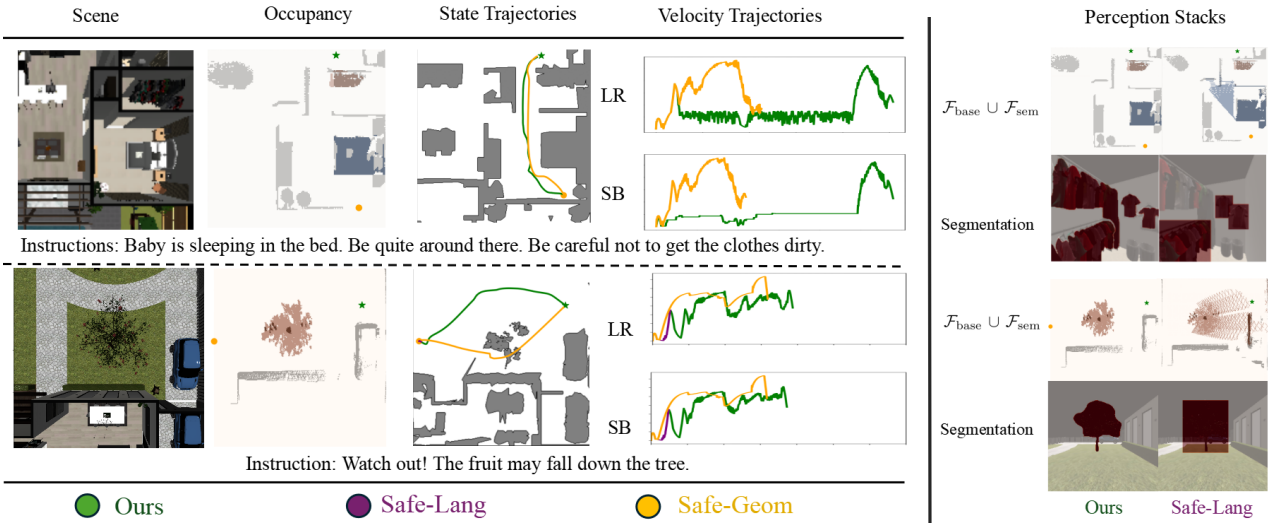



Fig. 2: Qualitative results. Left: our method follows language instructions and reaches the goal, while Safe-Geom violates semantic constraints. Right: Safe-Lang tends to produce overly conservative semantic failure sets, hindering task completion.

**Algorithm 1:** Sampling MPC-Based Safety Score Estimation (SB-MPC)

**Input:** initial state $x_{\text{init}}$, initial control guess $\mathbf{u}$;
**Output:** $V^{\text{safe}}$, safe control sequence found $\mathbf{u}^*$, $u^{\text{🛡}}$ ;
**Parameters:** number of iterative refinement steps $R$, number of control samples $N$, sampling variance $\sigma$;
**foreach** $r = 1 : R$ **do**
  **foreach** $n = 1 : N$ **do**
    $\tilde{\mathbf{u}}^n \sim \text{Gaussian}(\mathbf{u}, \sigma^2)$ ;
    $x_0^n \leftarrow x_{\text{init}}$ ;
    **for** $h = 0 : H-1$ **do**
      $x_{h+1}^n = f(x_h^n, \tilde{\mathbf{u}}_h^n)$ ;
    **end**
    $J^n = \min_{h=0,1,\dots,H} l(x_h^n)$ ;
  **end**
  $n^* = \arg\max_n J^n$ ;
  $\hat{V} \leftarrow J^{n^*}$ ;
  $\mathbf{u} \leftarrow \tilde{\mathbf{u}}^{n^*}$ ;
**end**
**Return** $\mathbf{u}^* \leftarrow \mathbf{u}$, $u^{\text{🛡}} \leftarrow \mathbf{u}_0$, $V^{\text{safe}} \leftarrow \hat{V}$ ;

safety score of the current state under the nominal control. The overall filtered control is given by:

$$u_{\text{filtered}} = \begin{cases} u_{\text{nom}}^k, & \text{if } V_{\text{nom}}^{\text{safe}} > \epsilon \\ u^{\text{🛡}}, & \text{otherwise} \end{cases} \tag{3}$$

where $u^{\text{🛡}}$ is the optimal safe control obtained from SB-MPC.

*Remark 1:* A warm-start sequence $\mathbf{u}^{\text{init}}$ is essential in marginally safe states, where stochastic sampling may otherwise miss existing safe trajectories. We therefore roll forward the previously optimized control sequence as the next initial guess.

**Smooth-Blending Safety Filter.** Rather than switching abruptly to a backup trajectory, the smooth-blending filter searches locally around $u_{\text{nom}}^k$ for the closest safe control, falling back to the SB-MPC trajectory if none exists. Specifically, we begin with a library of primitive control actions $\mathcal{U}_{\text{prim}}$. For each candidate action $u_{\text{prim}} \in \mathcal{U}_{\text{prim}}$, we evaluate its safety score $V_{\text{prim}}^{\text{safe}}$. We then retain only those actions with a positive $V_{\text{prim}}^{\text{safe}}$ that satisfy the CBF-inspired blending condition: $V_{\text{prim}}^{\text{safe}} - V^{\text{safe}} \geq -\gamma V^{\text{safe}}$, which ensures that the decrease in safety margin under the new action is bounded relative to the current safety score. Near the safety boundary ($V^{\text{safe}} \approx 0$) this prevents any further loss of margin, while deep inside the safe set it leaves more freedom for task performance. Among the retained primitives, we pick the one closest to the nominal control. This strategy yields smoother execution and has shown improved robustness to model mismatch in real-world experiments [5].

## V. Simulation Results

**Robot Dynamics.** We evaluate our framework in both simulated and hardware experiments. The robot is modeled as a 4D unicycle system with $x = [p_x, p_y, \theta, v]^\top \in \mathcal{X} \subseteq \mathbb{R}^4$, where $(p_x, p_y)$ denotes planar position, $\theta$ the heading angle, and $v$ the linear velocity. The control input is $u_t = [\omega, a]^\top \in \mathcal{U} \subseteq \mathbb{R}^2$, where $\omega$ and $a$ represent bounded angular velocity and linear acceleration, respectively. The discrete-time dynamics with a time step of $\Delta$ are given by:

$$\begin{aligned} p_{x,t+1} &= p_{x,t} + v_t \cos\theta_t \Delta, \\ p_{y,t+1} &= p_{y,t} + v_t \sin\theta_t \Delta, \\ \theta_{t+1} &= \theta_t + \omega_t \Delta, \\ v_{t+1} &= v_t + a_t \Delta. \end{aligned} \tag{4}$$

We impose bounds to ensure physically realistic motion: $0 \leq v_t \leq v_{\max}$, where $v_{\max}$ is set to $1.5\,\text{m/s}$. Angular velocity and acceleration are bounded by $|\omega| \leq 1\,\text{rad/s}$ and $|a| \leq 3\,\text{m/s}^2$, respectively.

**Simulation Environment.** We use the Habitat 3.0 simulator to evaluate our framework in diverse indoor and outdoor environments. We selected 12 environments, covering home-like scenes, public or semi-open spaces, and specialized

| Catagory | Target Constraints | | Success rate | Semantic safety rate | Timeout: perception | Timeout: safety filter |
|---|---|---|---|---|---|---|
| Exclusion Zones | 1. Avoid clothes (Bedroom)<br>2. Avoid plant corners (Courtyard)<br>3. Avoid rugs (House)<br>4. Avoid swimming pool (Poolside) | **Ours**<br>**Safe-Lang**<br>**Safe-Geom** | **90%**<br>25%<br>15% | **100%**<br>70%<br>25% | **5%**<br>45%<br>N/A | **5%**<br>0%<br>10% |
| Spatial Relationship Constraints | 1. Avoid going beneath trees (Yard)<br>2. Avoid going beneath the light (Living room)<br>3. Avoid going beneath the table (Dining hall)<br>4. Avoid going beneath ladders (House)<br>5. Avoid going under patio umbrellas (Garden) | | **92%**<br>80%<br>52% | **100%**<br>96%<br>80% | **0%**<br>16%<br>0% | **8%**<br>0%<br>28% |
| Buffer Zones-Based Constraints | 1. Maintain 0.25 m margin from cars (Garage)<br>2. Stay away from chairs (Living room) | | **80%**<br>N/A<br>0% | **100%**<br>N/A<br>0% | **10%**<br>N/A<br>N/A | **10%**<br>N/A<br>0% |
| Pace Modulation Constraints | 1. Move slower around the bed (Bedroom)<br>2. Move slower around cars (Garage)<br>3. Move slowly around chairs (Hallway)<br>4. Move slowly on the carpet (Dining hall)<br>5. Move slowly near the sofa (House)<br>6. Move slowly around bed and tub (Living room) | | **90%**<br>N/A<br>10% | **93%**<br>N/A<br>17% | **3%**<br>N/A<br>N/A | **0%**<br>N/A<br>7% |
| Hybrid Constraints | 1. Avoid table + move slowly on carpet (Dining hall)<br>2. Avoid ladders/rugs + move slowly near sofa (House)<br>3. Careful near stairs + stay away from chairs (Living room) | | **73%**<br>N/A<br>20% | **93%**<br>N/A<br>33% | **13%**<br>N/A<br>N/A | **7%**<br>N/A<br>13% |
| Intent-based constarints | 1. Be extra careful near stairs (Living room variant)<br>2. Fruits can fall from the trees (Yard) | | **60%**<br>N/A<br>0% | **100%**<br>N/A<br>0% | **20%**<br>N/A<br>N/A | **20%**<br>0%<br>0% |

Fig. 3: Quantitative simulation results. Our method demonstrates broad generality and consistently achieves the best performance across all baselines.

scenarios such as yards with swimming pools or gardens with trees. Each environment comes with language instructions testing exclusion zones, spatial relationships, pace modulation, intent-based, and hybrid constraints (listed in Fig. 3), each evaluated over five randomized trials. Instructions are issued at $t = 0$, trials terminate after 35 seconds, and we assume perfect state estimation and noise-free sensing in simulation.

**Implementation Details.** In all experiments, we use GPT-4o with a fixed initial prompt to parse language instructions into structured safety constraints. For perception, we use mask2former [26] for panoptic segmentation and ClipSeg [25] for open-vocabulary object recognition. We use $N = 50$ and $R = 2$ for SB-MPC-based safety score estimation. For least-restrictive filtering, we use $\epsilon = 0.25$, which corresponds to the robot's base radius. For the smooth blending filter, we use $\gamma = 5.0$ and 25 control primitives.

**Nominal Policy.** Our nominal policy consists of an A* planner followed by a low-level PD tracking controller without any safety penalty. For both simulation and hardware experiments, the A* planner operates at 2 Hz while the PD controller is at 20 Hz.

**Baselines.** We compare our approach with: 1) **Safe-Geom**: A geometric baseline where nominal actions are filtered to ensure safety with respect to $\mathcal{F}_{\text{base}}$. 2) **Safe-Lang** [12]: A method that updates safety representations using OWLv2 for open-vocabulary segmentation and maintains safety against both the semantic and geometric failure sets. Since its grid-based, warm-started reachability filter is real-time only for low-dimensional systems (typically $\leq$ 3D) [27], we replace it with our MPC-based safety filter for a fair comparison, avoiding safety degradation from computational latency.

**Evaluation Metrics.** We evaluate performance using four metrics. *Success rate* measures the percentage of scenarios in which the robot reaches the goal while satisfying both geometric and semantic safety constraints. *Semantic safety rate* captures the fraction of scenarios where semantic constraints are respected throughout the entire trajectory. We also report two types of failures: *Timeout:perception*, which occurs when overly conservative perception enlarges the semantic failure set and prevents goal completion, and *Timeout:safety filter*, which arises when the safety filter persistently overrides the nominal policy, causing failure to reach the goal despite available feasible paths.

### A. Overall Results

Results are summarized in Fig. 3 by constraint type. **Safe-Geom** cannot explicitly reason about semantics, resulting in a very low semantic safety rate and consequently a low overall success rate. **Safe-Lang** can handle exclusion-zone constraints but frequently generates overly conservative masks, as indicated by its particularly high *Timeout: Perception* metric. This blocks feasible paths, leading to a poor task completion rate. This effect is also evident in Fig. 2, where Safe-Lang over-segments the tree and eliminates safe passages. In contrast, our approach produces more accurate segmentation and robust occupancy integration, maintaining safety while avoiding unnecessary restrictions on motion.

In terms of other types of semantic constraints, while Safe-Lang lacks the capability to directly enforce spatial relationship constraints, such as "beneath", it projects the bounding box for the object specified in the instruction down to the ground, which inadvertently allows it to avoid such relational constraints. However, Safe-Lang, once again, leads to an overly conservative bounding box in this case, leading to a higher timeout rate compared to the proposed approach. Moreover, it is not able to handle other types of semantic constraints, e.g., based on buffer zones, pace modulation, and intent-based constraints. In contrast, our method is consistently able to maintain a high semantic safety rate, showing the effectiveness and generality of the proposed approach. Overall, these results highlight that our framework balances

semantic safety compliance with efficient task execution more effectively than existing approaches.

### B. LLM Parser Evaluation

We queried GPT-4o, with a fixed initial prompt, five times for each of the 12 simulation instructions. It produced correct safety constraints in 52 of 60 attempts without clarification, including zero-shot translation of ambiguous instructions such as *baby is sleeping in the bed. Be quiet around there*. In the remaining eight cases, the LLM expressed uncertainty and asked the user for clarification; all yielded the expected configurations after user feedback.

### C. Filter Evaluation

We isolate the safety filter from perception to evaluate it independently of grounding errors. Across all experiments, both filters maintained compliance with the specified failure sets; the semantic violations reported earlier stem from incorrect grounding of $\mathcal{F}_{\text{sem}}$, not from the filter. In stress tests in 10 highly cluttered environments (100 trials each, fixed nominal controller), zero collisions were observed across all 1,000 runs. We hypothesize that, for navigation systems, the effective safety value function converges within a short horizon because the robot can brake to a stop. This property enables near-recursive safety in practice with a relatively small MPC horizon $H$.

Among the two filtering schemes, the smooth-blending filter is particularly effective for kinematic modulation constraints (e.g., pace-limiting instructions), producing smoother velocity profiles and more natural behavior near constrained regions (see Fig. 2). For exclusion-zone tasks, performance differences relative to the least-restrictive filter are less pronounced. The improved smoothness comes at the cost of increased latency ($10.74\pm0.23$ ms vs. $2.09\pm0.10$ ms; NumPy CPU implementation without GPU acceleration), although both variants remain suitable for real-time deployment.

## VI. Hardware Experiments

We deployed our framework on a TurtleBot 2 platform equipped with an Intel RealSense D415 RGB-D camera and a 2D LiDAR for localization and mapping. The robot dynamics used for safety filtering and MPC follow (4), with maximum forward velocity capped at $v_{\max} = 0.5\text{m/s}$.

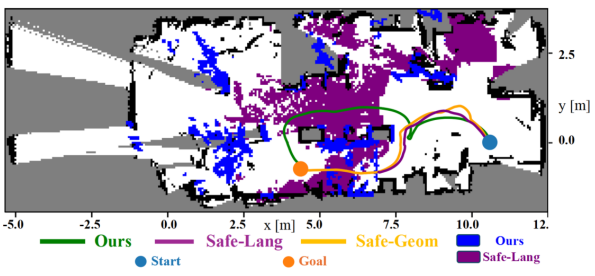



Fig. 4: Hardware experiment in a cluttered office. The robot reaches the goal while respecting semantic constraints with our method. Semantic failure sets from our method and Safe-Lang are shown in blue and purple, respectively.

### A. Spatial Constraint Grounding

We evaluate all methods in a cluttered office with chairs, desks, and monitors. The user issues the constraint $L =$ "*Don't go beneath the workstations*" at deployment time ($t = 0$). This requires reasoning about a spatial relationship not modeled in the geometric SLAM map (Safe-Lang projects bounding boxes to the ground by default, enabling a fair comparison). Safe-Geom avoided static obstacles and reached the goal but violated the semantic constraint by passing under the desk, while Safe-Lang produced an overly conservative semantic failure set and timed out. Our method avoided both physical and semantic violations. Despite odometry drift and noisy depth from occlusions and reflective monitor screens, it grounded the "desk" constraint reliably throughout execution.

### B. Extended Capability Demonstrations

We further evaluate real-world tasks that highlight capabilities unique to the proposed framework (videos on the project website[1]). In all scenarios, the framework maintains constraint compliance while completing the task: **(i) buffer zones in semi-dynamic scenes**, adapting in real time to a garbage bag dropped by a pedestrian and missed by 2D LiDAR SLAM; **(ii) behavioral constraints from vague instructions**, slowing down near the bed for *"Puppy is on the bed, try not to disturb her"*; **(iii) implicit spatial constraints**, identifying dense object clusters for *"avoid the messy area"*; **(iv) combined** spatial and behavioral constraints; and **(v) spoken instructions** issued at runtime.

## VII. Limitations and Future Work

In this work, we present a modular framework for language-conditioned safety in robot navigation, integrating (i) LLM-based specification synthesis, (ii) semantic perception grounding, and (iii) sampling-based safety filtering for real-time enforcement. Simulation and hardware experiments demonstrate that the framework can interpret diverse natural language instructions, ground them into actionable safety constraints, and enforce them during execution in cluttered environments.

Several limitations remain. First, missed detections or overly conservative 3D representations from segmentation errors and odometry drift can propagate into incorrect safety sets, and overly conservative buffers inferred from language may cause the robot to freeze; inherently vague instructions (e.g., "Don't interrupt the people talking face to face") also remain hard to formalize. Second, the filter assumes deterministic discrete-time dynamics; reasoning about dynamics uncertainty and probabilistic occupancy would improve robustness, particularly for systems such as quadrupeds. Finally, while this work focuses on navigation, manipulation tasks are often more semantic in nature. Extending language-conditioned safety to physical interaction—where constraints may involve fragile objects, force limits, or user preferences—presents a challenging but important direction for future research.

[1] https://annnymuser.github.io/Word2SafetyWebsite/

## REFERENCES


[1] B. Tearle, K. P. Wabersich, A. Carron, and M. N. Zeilinger, "A predictive safety filter for learning-based racing control," *IEEE Robotics and Automation Letters*, vol. 6, no. 4, pp. 7635–7642, 2021.

[2] R. M. Bena, G. Bahati, B. Werner, R. K. Cosner, L. Yang, and A. D. Ames, "Geometry-aware predictive safety filters on humanoids: From poisson safety functions to cbf constrained mpc," *arXiv*, 2025.

[3] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada, "Control barrier functions: Theory and applications," in *European Control Conference*, 2019, pp. 3420–3431.

[4] K. P. Wabersich, A. J. Taylor, J. J. Choi, K. Sreenath, C. J. Tomlin, A. D. Ames, and M. N. Zeilinger, "Data-driven safety filters: Hamilton-jacobi reachability, control barrier functions, and predictive methods for uncertain systems," *IEEE Control Systems Magazine*, vol. 43, no. 5, pp. 137–177, 2023.

[5] J. Borquez, K. Chakraborty, H. Wang, and S. Bansal, "On safety and liveness filtering using hamilton-jacobi reachability analysis," *IEEE Transactions on Robotics*, 2024.

[6] K.-C. Hsu, H. Hu, and J. F. Fisac, "The safety filter: A unified view of safety-critical control in autonomous systems," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 7, 2023.

[7] W. Yu, N. Gileadi, C. Fu, S. Kirmani, K.-H. Lee, M. G. Arenas, H.-T. L. Chiang, T. Erez, L. Hasenclever, J. Humplik *et al.*, "Language to rewards for robotic skill synthesis," *arXiv*, 2023.

[8] T. Xie, S. Zhao, C. H. Wu, Y. Liu, Q. Luo, V. Zhong, Y. Yang, and T. Yu, "Text2reward: Reward shaping with language models for reinforcement learning," *arXiv preprint arXiv:2309.11489*, 2023.

[9] P. Goyal, S. Niekum, and R. J. Mooney, "Using natural language for reward shaping in reinforcement learning," in *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 2019.

[10] J. Song, Z. Zhou, J. Liu, C. Fang, Z. Shu, and L. Ma, "Self-refined large language model as automated reward function designer for deep reinforcement learning in robotics," *arXiv*, 2023.

[11] 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.

[12] L. Santos, Z. Li, L. Peters, S. Bansal, and A. Bajcsy, "Updating robot safety representations online from natural language feedback," in *2025 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2025, pp. 7778–7785.

[13] L. Brunke, Y. Zhang, R. Römer, J. Naimer, N. Staykov, S. Zhou, and A. P. Schoellig, "Semantically safe robot manipulation: From semantic scene understanding to motion safeguards," *IEEE Robotics and Automation Letters*, 2025.

[14] Z. Liao, Y. Hu, J. Zhang, X. Qi, X. Zhang, and W. Wang, "So-slam: Semantic object slam with scale proportional and symmetrical texture constraints," *IEEE Robotics and Automation Letters*, 2022.

[15] M. Li, S. Liu, H. Zhou, G. Zhu, N. Cheng, T. Deng, and H. Wang, "Sgs-slam: Semantic gaussian splatting for neural dense slam," in *European Conference on Computer Vision*, 2024.

[16] S. Zhu, R. Qin, G. Wang, J. Liu, and H. Wang, "Semgauss-slam: Dense semantic gaussian splatting slam," *arXiv*, 2024.

[17] Q. Gu, A. Kuwajerwala, S. Morin, K. M. Jatavallabhula, B. Sen, A. Agarwal, C. Rivera, W. Paul, K. Ellis, R. Chellappa *et al.*, "Conceptgraphs: Open-vocabulary 3d scene graphs for perception and planning," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 5021–5028.

[18] A. Takmaz, E. Fedele, R. W. Sumner, M. Pollefeys, F. Tombari, and F. Engelmann, "Openmask3d: Open-vocabulary 3d instance segmentation," *arXiv preprint arXiv:2306.13631*, 2023.

[19] W. He, S. Jamonnak, L. Gou, and L. Ren, "Clip-s4: Language-guided self-supervised semantic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 11 207–11 216.

[20] Z. Zhou, Y. Lei, B. Zhang, L. Liu, and Y. Liu, "Zegclip: Towards adapting clip for zero-shot semantic segmentation," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2023, pp. 11 175–11 185.

[21] I. Vizzo, T. Guadagnino, J. Behley, and C. Stachniss, "Vdbfusion: Flexible and efficient tsdf integration of range sensor data," *Sensors*, vol. 22, no. 3, p. 1296, 2022.

[22] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, "Control barrier function based quadratic programs for safety critical systems," *IEEE Transactions on Automatic Control*, 2016.

[23] 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)*, 2019, pp. 1758–1765.

[24] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Aggressive driving with model predictive path integral control," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016, pp. 1433–1440.

[25] T. Lüddecke and A. Ecker, "Image segmentation using text and image prompts," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2022, pp. 7086–7096.

[26] B. Cheng, I. Misra, A. G. Schwing, A. Kirillov, and R. Girdhar, "Masked-attention mask transformer for universal image segmentation," 2022.

[27] 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," *arXiv preprint*, 2019.