2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Context Matters! Relaxing Goals with LLMs for Feasible 3D Scene Planning

Emanuele Musumeci[1,*], Michele Brienza[1,*], Francesco Argenziano[1,*], Abdel Hakim Drid[2], Vincenzo Suriani[1], Daniele Nardi[1], and Domenico D. Bloisi[3]

***Abstract*— Embodied agents need to plan and act reliably in real and complex 3D environments. Classical planning (e.g., PDDL) offers structure and guarantees, but in practice it fails under noisy perception and incorrect predicate grounding. On the other hand, Large Language Models (LLMs)-based planners leverage commonsense reasoning, yet frequently propose actions that are unfeasible or unsafe. Following recent works that combine the two approaches, we introduce ContextMatters, a framework that fuses LLMs and classical planning to perform hierarchical goal relaxation: the LLM helps ground symbols to the scene and, when the target is unreachable, it proposes functionally equivalent goals that progressively relax constraints, adapting the goal to the context of the agent's environment. Operating on 3D Scene Graphs, this mechanism turns many nominally unfeasible tasks into tractable plans and enables context-aware partial achievement when full completion is not achievable. Our experimental results show a +52.45% Success Rate improvement over state-of-the-art LLMs+PDDL baseline, demonstrating the effectiveness of our approach. Moreover, we validate the execution of ContextMatters in a real world scenario by deploying it on a TIAGo robot. Code, dataset, and supplementary materials are available to the community at https://lab-rococo-sapienza.github.io/context-matters/.**

I. Introduction

Planning with robots and embodied agents requires generating action sequences that an agent must execute to achieve a goal within its environment. This process becomes more challenging when moving from simulated to real-world settings, where robots can fail due to a failure in correctly capturing the world model in the plan's preconditions. The two most common complementary task planning approaches can both break. Pure LLM planners [2], [3], [4] leverage their commonsense knowledge to interpret intent [5], but often hallucinate missing preconditions and actions, yielding optimistic sequences that collapse at execution. On the other hand, pure PDDL planners [6] provide guarantees, yet treat unmet preconditions as dead-ends: if the goal, as represented in the planning domain, is unsatisfiable, planning fails, offering no principled way to adapt the goal while preserving intent.

Taking for instance a task "*set the dining table with two forks*", carried out within a 3D environment, typically represented as a 3D Scene Graph (3DSG) [1], [7], containing a dining table, cutlery drawers, a shelf, and a running dishwasher. In the actual scene, the top drawer is blocked and no clean forks are available, while spoons are reachable

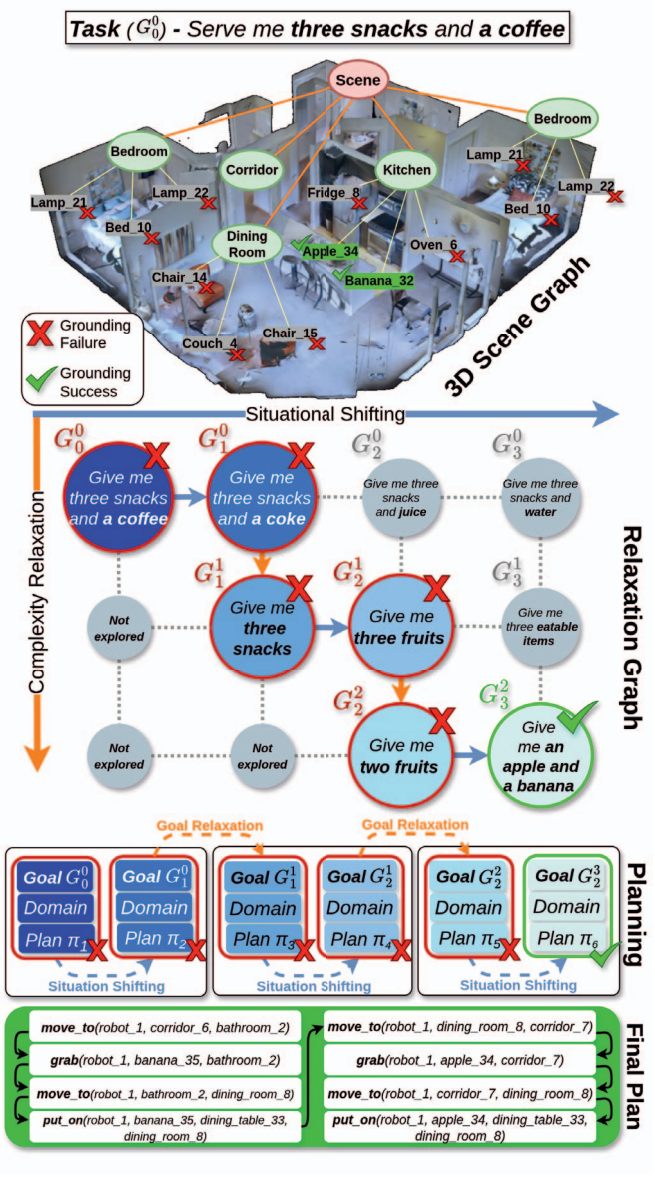

Fig. 1. Our architecture takes as input a 3D Scene Graph [1] of the environment and a task expressed in natural language. Unfeasible goals can be relaxed up to a certain degree into semantically similar ones, computing the corresponding plan. This mimics the capability of humans to change expected outcomes on-the-go depending on the *context*.

[1]Department of Computer, Automation and Management Engineering, Sapienza University of Rome, Rome, Italy lastname@diag.uniroma1.it [2]Department of Electronics and Automation, Mohamed Khider University of Biskra, Biskra, Algeria abdelhakim.drid@univ-biskra.dz [3]International University of Rome UNINT, Rome, Italy domenico.bloisi@unint.eu
*Authors contributed equally.

on the shelf. An LLM-only pipeline would often propose the idealized sequence "open drawer → pick forks → place", unless the blockage and unavailability are explicitly grounded; a PDDL-only pipeline correctly encodes those constraints and returns *no plan*. Both these scenarios lack a mechanism treating failure as a cue to modify both *what* to achieve and *where/how* to achieve it, by adapting the user's intent to the *context* of the scene without discarding it.

We ask therefore the following research question: instead of failing, can an agent intelligently analyze its 3D environment to relax the goal into *functionally equivalent but contextually achievable objectives*?

To this end, we introduce *ContextMatters*, a bidimensional relaxation architecture that jointly searches over *functionality* (*what to* achieve, at varying levels of semantic equivalence) and *feasibility* (*where/how* to achieve it under symbolic and physical constraints) to convert failure signals into intent-preserving, executable goals. Consider the example depicted in Fig. 1, where the user requests "*serve me three snacks and a coffee*" in a home represented as a 3D Scene Graph. The scene makes the original specification difficult since some beverages are unavailable while only a subset of snacks (*e.g.,* `banana_32`, `apple_34`) are reachable. *ContextMatters* explores nearby goal variants such as "*three snacks and water/juice*", "*two fruits*", or "*three edible items*", and validates each candidate with a classical planner until a concrete plan emerges. By employing a bidimensional relaxation, *ContextMatters* proposes the *minimal* scene-grounded modification that preserves intent while ensuring preconditions hold, converging on an executable plan when a relaxed goal satisfies feasibility checks. By comparing our approach with state-of-the-art LLM+PDDL planners on 3DSGs, we show the effectiveness of our architecture achieving +52.45% improvement, while also executing it on a real robot setup. To support research in this direction, we also provide a dataset of tasks that require some degree of relaxations in order to be correctly carried out in the environment.

To summarize, our main contributions are:

- a novel contextual goal-relaxation formalism that reasons along two axes (functionality and feasibility) to preserve user intent while yielding executable goals.
- *ContextMatters*, a planning framework that couples LLM commonsense for goal proposal with classical planning for feasibility validation and plan synthesis.
- a new dataset of 141 relaxation-prone tasks compatible with popular 3D environments and 3DSGs [1], [8].
- an empirical evaluation on 3DSG planning benchmarks and a real-world demonstration on a TIAGo robot.

The remainder of this paper is structured as follows. Section II reviews related work on robot planning using LLMs and classical methods, while Section III details the proposed approach. Experimental results are presented in Section IV, and conclusions are drawn in Section V.

II. Related Work

3D Scene Graphs. 3D Scene Graphs [1], [7] have recently emerged as a versatile representation for indoor and outdoor environments [9], [10]. Graph nodes typically refer to scene objects and their attributes (like materials, or affordances); graph edges denote spatial and semantic relations between the primitives of the environment. Their compact structure allows to prompt a whole scene to an LLM for querying, facilitating a variety of downstream applications, like navigation in the environment [11], manipulation [12], and task planning [13], [14].

LLMs as Planners. LLMs are increasingly used for planning in embodied agents. In the seminal SayCan framework [15], a robot couples its observations and affordances with an LLM to ground high-level tasks in real settings, exploiting the model's semantic priors. Subsequent work uses LLMs as planners: with careful prompting, they function as zero-shot planners [16], [17]; with lightweight few-shot fine-tuning, they outperform state-of-the-art Vision-Language Navigation (VLN) models trained on larger datasets, benefiting from intrinsic commonsense [17], [18]. Performance improves further when inputs adopt structured representations, such as 3DSGs, which inject additional semantic context to the scene [13]. Advances in open-vocabulary perception have broadened real-world applicability by enabling flexible grounding of novel objects and scenes [19], [20]. Finally, multi-level frameworks integrate reasoning, planning, and motion generation within a single LLM-centric loop [21]. Despite these improvements, robust long-horizon planning remains still a key open challenge.

LLMs and Classical Planning. Combining LLMs with classical planners enables solving long-horizon tasks more reliably than pure LLM planning. For example, Ding et al. [22] dynamically augment the robot's action knowledge and modify the preconditions and effects of actions by leveraging task-oriented commonsense knowledge. However, these approaches struggle in completely unseen environments, as they rely on a strict definition of both domain and problem, often leading to failures in these cases. To combine the strengths of natural language reasoning and classical planning, Liu et al. [23] propose a method that translates a natural language description of a planning problem into a PDDL representation, allowing it to be solved using a PDDL planner. Since LLMs can still hallucinate and generate infeasible plans due to incomplete domain knowledge, Liu et al. [24] introduce DELTA, an LLM-guided task planning framework. DELTA leverages 3D Scene Graphs as environmental representations within LLMs to quickly generate planning problem descriptions. To further improve planning efficiency, it breaks down long-term task goals into an autoregressive sequence of subgoals using LLMs, enabling automated planners to solve complex tasks more effectively.

DELTA is the current state-of-the-art in combining LLMs with structured scene representations for task planning. Its main limitation is its exclusive reliance on LLMs to produce a directly usable symbolic domain. Generation inaccuracies in this phase, such as invalid terms, predicates or grounding mismatches, can prevent producing plans that are not executable in the modeled environment. *ContextMatters* addresses these issues through iterative domain validation, using a symbolic PDDL validator to provide feedback for its refinement.

III. METHODOLOGY

This section is organized as follows. Section III-A provides a formal definition of our planning approach, and introduces the *shifting* and *relaxation* operators. Section III-B highlights the architecture of our context-augmented planning system. Lastly, Section III-C describes the implementation of the system.

A. Contextual Task Adaptation

Stateful domain specification: We use a PDDL-like *stateful domain* $\Sigma = \langle \mathit{Obj},\ \mathit{Pred},\ \mathit{Act},\ \mathit{Init} \rangle$, where:

- *Obj:* finite set of *typed object constants* (*e.g.,*, `cup_1:cup`, `table_3:table`, `kitchen:room`);
- *Pred*: set of *predicate symbols* with arities and type signatures; may be partitioned into *static* (never change) and *fluent* (state-dependent) predicates, and may include *derived predicates*;
- *Act*: set of *action schemas* (operator templates) with parameters, preconditions, and effects;
- *Init*: the *initial state* at $t{=}0$, a set of true ground atoms over *Obj* and *Pred*.

Representation mapping: Given as input a 3DSG of a scene $\mathcal{S}_{\text{sem}}$, the function $\mathcal{M}_{\text{repr}}$ maps $\mathcal{S}_{\text{sem}}$, a knowledge base $\mathcal{K}$, and an action library $\mathcal{L}_{\text{act}}$ to a domain specification:

$$\mathcal{M}_{\text{repr}}(\mathcal{S}_{\text{sem}}, \mathcal{K}, \mathcal{L}_{\text{act}}) \ \mapsto\ \Sigma = \langle \mathit{Obj}, \mathit{Pred}, \mathit{Act}, \mathit{Init} \rangle.$$

In practice: *Obj* is derived from typed instances in $\mathcal{S}_{\text{sem}}$; *Pred* from relations/attributes plus $\mathcal{K}$; *Act* from $\mathcal{L}_{\text{act}}$; *Init* from facts extracted/derived from $\mathcal{S}_{\text{sem}}$ and $\mathcal{K}$. From now on, $\mathcal{M}_{\text{repr}}(\mathcal{S}_{\text{sem}})$ will be shorthand for $\mathcal{M}_{\text{repr}}(\mathcal{S}_{\text{sem}}, \mathcal{K}, \mathcal{L}_{\text{act}})$.

From scene to planning problem: Given a semantic environment $\mathcal{S}_{\text{sem}}$ and an initial goal G_0, the planning problem is then defined as the tuple

$$\mathcal{P}_0 \ =\ \langle \Sigma_0,\ G_0 \rangle,$$

where $\Sigma_0 \ =\ \mathcal{M}_{\text{repr}}\big(\mathcal{S}_{\text{sem}}\big) \ =\ \langle \mathit{Obj}_0,\ \mathit{Pred}_0,\ \mathit{Act},\ \mathit{Init}_0 \rangle$ represents the initial planning domain specification induced by the environment.

Situational shifts: The *situational shift operator*

$$\Sigma_k \ =\ \Gamma_{\text{shift}}\big(\Sigma_{k-1},\, G_0,\, \mathcal{S}_{\text{sem}},\, \mathcal{M}_{\text{repr}}\big), \quad k \geq 1,$$

adapts the agent's understanding of the operating environment to the goal G_0. Assuming the set of actions *Act* does not change, each shifted specification has the form

$$\Sigma_k \ =\ \langle \mathit{Obj}_k,\ \mathit{Pred}_k,\ \mathit{Act},\ \mathit{Init}_k \rangle.$$

The components Obj_k and $Init_k$ are adapted to be relevant for the shifted planning problem after the shifting. We denote the k-fold application of the situational shift by $\Gamma^k_{\text{shift}}\big(\Sigma_0,\, G_0,\, \mathcal{S}_{\text{sem}},\, \mathcal{M}_{\text{repr}}\big)$. Similar, a *goal-shift operator*

$$G_k \ =\ \Gamma_{\text{goal}}\big(G_{k-1},\, \mathcal{S}_{\text{sem}},\, \mathcal{M}_{\text{repr}}\big), \quad k \geq 1,$$

produces reformulations of the original intent over the current domain vocabulary. We write $G_k \sim G_0$ when G_k preserves the intent of G_0 under the vocabulary of Σ_k, *i.e.,*, $\Sigma_k \models G_k$ and G_k is logically interchangeable with G_0 up to object renaming and available predicates induced by $\mathcal{M}_{\text{repr}}$.

Relaxation operator: Fix a domain specification Σ (*e.g.,* Σ_0 or the current Σ_k). For goal formulas interpreted over Σ, let $\mathrm{Mod}_\Sigma(G) = \{\, s \mid s \models G \,\}$ denote the set of states that satisfy G. We define the *relaxation preorder* $\succeq_{\text{rel}}$ by

$$G' \succeq_{\text{rel}} G \quad \Longleftrightarrow \quad \mathrm{Mod}_\Sigma(G) \subseteq \mathrm{Mod}_\Sigma(G').$$

We write $G' \succ_{\text{rel}} G$ when the inclusion is strict. Intuitively, G' is a *weaker / more general / more abstract* goal than G. A *relaxation operator*

$$G^i \ =\ \Delta_{\text{rel}}\big(G^{i-1}, \Sigma\big), \quad i \geq 1,$$

is *valid* if it is monotone w.r.t. $\succeq_{\text{rel}}$, *i.e.,* $G^i \succeq_{\text{rel}} G^{i-1}$. By iterating, it induces a hierarchy

$$G^m \succeq_{\text{rel}} \cdots \succeq_{\text{rel}} G^1 \succeq_{\text{rel}} G^0,$$

or, in strict form when each step truly relaxes the goal,

$$G^m \succ_{\text{rel}} \cdots \succ_{\text{rel}} G^1 \succ_{\text{rel}} G^0.$$

We denote the i-fold application of the relaxation operator by $\Delta^i_{\text{rel}}(G^0, \Sigma)$.

According to these definitions, some relaxation examples are: (i) drop conjuncts (from $\wedge$-goals); (ii) replace constants with types or sets (*e.g.,* `cup_3` $\rightarrow \exists x{:}$`cup`); (iii) generalize predicates via $\mathcal{K}$ (*e.g.,* `on` $\rightarrow$ `supported_by`); (iv) widen numeric thresholds; (v) introduce disjunctions. Each increases $\mathrm{Mod}_\Sigma(\cdot)$ and thus respects $\succeq_{\text{rel}}$.

Combining the two operators: The application of a *shift* Γ^k_{shift} followed by a *relaxation* Δ^i_{rel} to the initial pair $\langle \Sigma_0, G^0_0 \rangle$ yields a shifted planning domain Σ_k and a shifted-relaxed goal G^i_k, obtained by adapting the agent's understanding of the operating environment and then relaxing the goal within the shifted domain. Formally, we first compute the k-fold shift

$$\Sigma_k \ =\ \Gamma^k_{\text{shift}}\big(\Sigma_0,\, G^0_0,\, \mathcal{S}_{\text{sem}},\, \mathcal{M}_{\text{repr}}\big),$$

(optionally aligning the goal to the new vocabulary as $G^0_k = \Gamma^k_{\text{goal}}(G^0_0, \mathcal{S}_{\text{sem}}, \mathcal{M}_{\text{repr}})$, otherwise $G^0_k \equiv G^0_0$), and then build the i-fold relaxation *within* Σ_k:

$$G^i_k \ =\ \Delta^i_{\text{rel}}\big(G^0_k,\, \Sigma_k\big), \qquad i \geq 0.$$

We can now form a two-dimensional family of planning problems by combining each shifted domain Σ_k and its relaxed goals G^i_k, whose solution is a plan $\pi_{i,k}$:

$$\mathcal{P}_{i,k} \ =\ \langle \Sigma_k,\, G^i_k \rangle, \qquad \forall\, k \in \{0, \ldots, N\},\ \forall\, i \in \{0, \ldots, M\}.$$

Hence, each $\mathcal{P}_{i,k}$ is a distinct planning problem with (i) domain specification Σ_k and (ii) goal specification G^i_k; a plan $\pi_{i,k}$ must satisfy G^i_k *within* Σ_k.

Relaxation graph: For each pair (i, k) we consider the shifted domain Σ_k and the shifted–relaxed goal G^i_k, yielding

$$\mathcal{P}_{i,k} \ =\ \langle \Sigma_k,\, G^i_k \rangle, \qquad \pi_{i,k} \ =\ \mathrm{Plan}\big(\Sigma_k,\, G^i_k\big).$$

We define the *relaxation graph* as the directed graph

$$\mathcal{G}_{\text{relax}} \ =\ (V, E), \quad V = \{\mathcal{P}_{i,k} \mid i = 0{:}M,\ k = 0{:}N\},$$

with horizontal (shift) edges

$$E_{\text{shift}} = \{\, \mathcal{P}_{i,k} \rightarrow \mathcal{P}_{i,k+1} \mid i, k \,\},$$

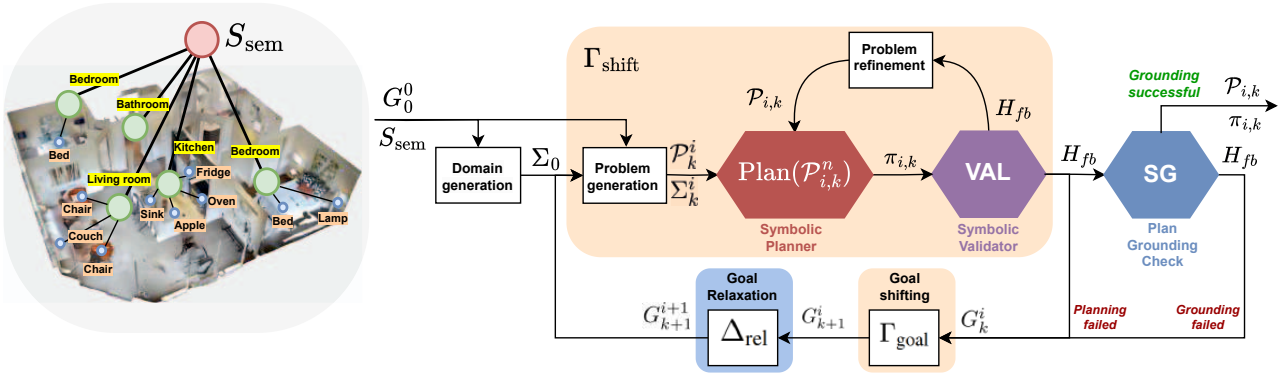

Fig. 2. Proposed architecture. The generated domain is followed by a planning attempt. In case of insuccess, domain refinement follows the feedback of a symbolic validator (purple hexagon) and a grounding verifier (blue hexagon), mapping domain elements to real objects in the 3DSG. If the refinement loop is unsuccessful, an attempt is made to *shift* the goal (orange block) and to optionally *relax* it (blue block) to a more feasible but still intent-preserving version.

and vertical (relax) edges

$$E_{\text{rel}} = \{\, \mathcal{P}_{i,k} \to \mathcal{P}_{i+1,k} \mid i,k \,\},$$

such that $E = E_{\text{shift}} \cup E_{\text{rel}}$, and root $\mathcal{P}_{0,0} = \langle \Sigma_0, G_0^0 \rangle$. Any path in $\mathcal{G}_{\text{relax}}$ encodes a sequence of shifts and relaxations explored in search of a solvable instance. Visually:

$$\begin{aligned}
&\langle \Sigma_0, G_0^0 \rangle \;\to\; \langle \Sigma_1, G_1^0 \rangle \;\to\; \cdots \;\to\; \langle \Sigma_N, G_N^0 \rangle, \\
&\langle \Sigma_0, G_0^1 \rangle \;\to\; \langle \Sigma_1, G_1^1 \rangle \;\to\; \langle \Sigma_2, G_2^1 \rangle \;\to\; \cdots, \\
&\quad\vdots \\
&\langle \Sigma_0, G_0^M \rangle \;\to\; \langle \Sigma_1, G_1^M \rangle \;\to\; \cdots \;\to\; \langle \Sigma_N, G_N^M \rangle.
\end{aligned}$$

Horizontally (increasing k), Γ_{shift} adapts the domain to the scene while preserving the original intent; vertically (increasing i), Δ_{rel} weakens the goal within the fixed Σ_k. We traverse this space by prioritizing first moving right (shifting) as long as planning shows progress (*e.g.,* more preconditions become satisfiable or search effort drops); if progress stalls, we step down (relax) to simplify the goal, repeating until some $\mathcal{P}_{i,k}$ is solvable, returning $\pi_{i,k}$. Because Σ_k is derived from the 3DSG, the plan is grounded to the robot's percepts.

B. Architecture

In order to implement the *shift operators* Γ_{shift} and Γ_{goal}, producing respectively a new shifted domain and goal, and the *relaxation operator* Δ_{rel}, producing a relaxed goal, the architecture (see Fig. 2) exploits commonsense reasoning of LLMs, through the intrinsic representation $\mathcal{M}_{\text{repr}}$. The agent's operating environment is initially modeled as a 3DSG $\mathcal{S}_{\text{sem}}$, then filtered by keeping only information about the location of objects in the scene, each tagged with a brief description.

Domain generation: A symbolic domain Σ_0, based on the agent's skills and the environment, can be either provided as a structured prior, or generated from $\mathcal{S}_{\text{sem}}$, the initial goal G_0, a knowledge base $\mathcal{K}$ and the action library $\mathcal{L}_{\text{act}}$.

Iterative problem refinement: The Γ_{shift} operator generates a shifted planning problem $\mathcal{P}_{i,k}$ from the domain Σ_0, the current goal G_k^i and the 3DSG S_{sem}. $\mathcal{P}_{i,k}$ might not be immediately solvable, due to hallucinations introduced in the domain generation or misalignments between any element of the tuple $\langle Obj, Pred, Act, Init \rangle$ and the ground information contained in $\mathcal{S}_{\text{sem}}$. This would make impossible to produce an executable plan, namely a plan $\pi_{i,k}$ that is groundable in the scene graph $\mathcal{S}_{\text{sem}}$. Therefore, through *Problem refinement*, the generated problem is corrected maximizing its semantic relevance to the current goal and its alignment to $\mathcal{S}_{\text{sem}}$. The refinement is iterated in case a solution plan is not found. The set H_{fb}, containing natural language feedback obtained throughout the symbolic planning process, is used to guide the refinement, until a plan can be computed. We can consider the *situational shifting operator* Γ_{shift}, as the composition of problem generation and iterative refinement. At this stage, the goal is not shifted yet. The planning process will conservatively attempt to modify the representation of the operating environment without modifying the goal, if possible, to avoid any unnecessary variation with respect to the original task.

Plan grounding check: If a successful plan $\pi_{i,k}$ is obtained, its correctness is evaluated by verifying the grounding of each action parameter, in an attempt to find a mapping between elements in domain Σ_k^i and the environment representation S_{sem}. If this process is successful, the plan is accepted.

Goal shifting and relaxation: If even after the iterative refinement it is still impossible to compute a groundable plan $\pi_{i,k}$ for the current planning problem $\mathcal{P}_{i,k}$, the *situational shifting* proceeds with a *goal shifting* step, followed by a *goal relaxation* $G_{k+1}^{i+1} = (\Gamma_{\text{goal}}^k \circ \Delta_{\text{rel}})(G_k^i)$, producing a relaxed planning problem $\mathcal{P}_{i+1,k+1}$. The generation, refinement, grounding and relaxation process is then iteratively repeated, until a groundable plan is computed, or a maximum number of attempts has been reached. As shown in Fig. 2, the *shifting operator* Γ_{shift} is the composition of the domain generation, problem shifting sub-step and goal shifting sub-steps.

C. Implementation

Algorithm 1 describes the high-level steps documented in Section III-B. In practice, they are realized as a set of LLM prompts, translating the abstract architecture into concrete instructions, containing **constraints**, **data** or **examples**. Full prompts are available in the provided repository.

The raw 3DSG undergoes *Context Distillation* in all LLM prompts, where it is pre-processed into a distilled semantic representation S_{sem}, optimizing the LLM context window

Algorithm 1: *Implementation of the architecture.*

Input: G_0^0; Semantically enhanced 3DSG $\mathcal{S}_{\text{sem}}$; Knowledge base $\mathcal{K}$; Action Library $\mathcal{L}_{\text{act}}$
Output: Executable plan $\pi_{i,k}$; Final problem $\mathcal{P}_{i,k}$

$\Sigma_0 \leftarrow LLM(G_0, \mathcal{S}_{\text{sem}}, \Sigma_{\text{desc}})$ ▷ Domain generation
$i, k \leftarrow 0$; $groundOK \leftarrow False$;
while $\pi_{i,k} = \emptyset \wedge \neg groundOK$ **do**
  ▷ Γ_{shift}
  $\mathcal{P}_{i,k} \leftarrow LLM(\Sigma_k^i, G_k^i, \mathcal{S}_{\text{sem}})$; ▷ Probl. gen.
  $\pi_{i,k} \leftarrow \text{Plan}(\mathcal{P}_{i,k})$; ▷ Symbolic Planner
  while $\pi_{i,k} = \emptyset$ **do**
    $H_{fb} \leftarrow \text{VAL}(\mathcal{P}_{i,k})$; ▷ Symbolic Validator
    $\mathcal{P}_{i,k} \leftarrow LLM(G_k^i, \Sigma_k^i, \mathcal{S}_{\text{sem}}, H_{fb})$; ▷ Probl. ref.
    $\pi_{i,k} \leftarrow \text{Plan}(\mathcal{P}_{i,k})$;
  if $\pi_{i,k} \neq \emptyset$ **then**
    groundOK, $H_{fb} \leftarrow \text{SG}(\pi_{i,k}, \mathcal{S}_{\text{sem}})$; ▷ Grounding
    if *groundOK* **then**
      break;
  $G_{k+1}^i \leftarrow LLM(G_k^i, \mathcal{S}_{\text{sem}})$; ▷ Γ_{goal}
  $G_{k+1}^{i+1} \leftarrow LLM(G_{k+1}^i, \mathcal{S}_{\text{sem}})$; ▷ Δ_{rel}
  $i \leftarrow i+1$; $k \leftarrow k+1$
return $\mathcal{P}_{i,k}$, $\pi_{i,k}$

usage, by retaining only relevant data. Algorithm 1 comprises two nested loops: the outer loop generates relaxed goals and candidate plans; the inner loop refines and validates them, or, if unsuccessful, triggers the next relaxation/shift.

Domain generation: Initially, the **planning domain** Σ_0 is generated, by instructing the LLM to create a PDDL domain satisfying the initial goal G_0^0 in $\mathcal{S}_{\text{sem}}$, with the prompt:

> **System Prompt:** Given a description of the planning domain, the domain actions and the domain objects, you must generate a PDDL domain file. **[Generation constraints]**, **[PDDL 1.2 Specifications]**, **[Domain description]**, **[PDDL domain example]**
>
> **User Prompt:** Extract object types and actions from the following description and generate a corresponding PDDL domain. **[Goal]**. **[Domain description]**. The generated PDDL domain incorporates these elements and respects the provided preconditions and effects. **[Generation constraints]**

At every outer loop iteration, the LLM generates a PDDL problem matching the generated PDDL domain, as follows:

> **System Prompt:** Generate a PDDL problem file given: **[Description of the provided elements]**. The environment is represented as a scene graph, with the following features: **[3DSG structure description]**. **[PDDL syntax constraints]**.
>
> **User Prompt:** **[Goal]**, **[PDDL domain]**, **[Distilled 3DSG]**, **[3DSG example]**, **[PDDL domain/problem example]**

Iterative problem refinement: If a plan cannot be found, the solution is validated by symbolic validator, producing a set of natural language feedback H_{fb} about PDDL correctness. The LLM is then prompted to reason about the cause of the error in a chain-of-thought, to correct the PDDL:

> **System Prompt:** Given the PDDL domain and problem, planner output, and scene, your job is to figure out why planning failed.
>
> **User Prompt:** **[PDDL domain]**, **[PDDL problem]**, **[Goal]**, **[Distilled 3DSG]**, **[Planning/Validation/Grounding feedback]**. Please provide a clear explanation of the possible reason(s) for the planning failure. At the end provide detailed suggestions to solve the issues you found. **[Generation constraints]**

The LLM is then instructed to properly correct the PDDL problem by integrating the analysis produced in the reasoning step and the feedback from the validation H_{fb}. while adhering to a subset of the PDDL 1.2 formalism [6].

> **System Prompt:** Given the PDDL domain, the previous PDDL problem, and a failure analysis, rewrite or fix the PDDL problem to address the failure according to the given suggestions.
>
> **User Prompt:** **[PDDL problem]**, **[Previous LLM output]**. (1) Fix domain-problem inconsistencies. (2) Math PDDL objects/-types/predicates with the domain and (3) Ensure the goal is achievable. **[Output format]**. **[Generation constraints]**.

The two-step strategy improves the chances of success.

Grounding check: The correctness of a plan is verified by virtually executing the plan in the 3DSG, to detect LLM-induced hallucinations, in case objects not available in the original 3DSG are misplaced in the generated PDDL domain. In practice, this consists in a scene consistency check over the candidate symbolic plan: plans introducing objects having no correspondence in the actual 3DSG are rejected. Feedback obtained from this additional check is then added to the set H_{fb}, to correct the problem in subsequent iterations.

Goal shifting and relaxation: If a plan is not found, the goal must be reformulated through shifting and relaxation. The goal shifting step Γ_{goal}, prompts the LLM to reformulate the goal using alternative objects in the scene.

> **System Prompt:** Identify the objects necessary to perform a task in a scene graph. Given a high-level task, your goal will be to identify similar objects, objects that can both be used for the same functions. **[Execution example]**
>
> **User Prompt:** **[Distilled 3DSG]**, **[Goal]**

The goal relaxation step Δ_{rel}, is implemented by instructing the LLM to reason about how to decompose the goal in single steps and to remove implicit restrictions, if needed, or retain the previous formulation is already feasible, with the prompt:

> **System Prompt:** Given a task and a description of the available objects and their locations, determine whether the given objective is achievable or propose a relaxed objective, semantically similar to the original and still feasible, removing the least important restrictions first. **[Generation constraints]** **[Execution example]**
>
> **User Prompt:** **[Previous LLM response]**, **[Goal]**.

As both steps are generative in nature, they can be considered LLM-based "proposal" functions. Consequently, an invocation of a shifting or relaxation does not necessarily imply a modification of the goal; it represents an attempt to relax the objective conditioned on the available context.

IV. Experimental Results

In this section we discuss our experimental setup comprising baselines, dataset, metrics and main results (Sections IV-A, IV-B, IV-C and IV-D). Additionally, we also demonstrate our approach on a real robot setup (Section IV-F).

A. Baselines

We compare against three popular baselines: LLMAsPlanner, SayPlan, and DELTA.

LLMAsPlanner is a simple prompting baseline that provides the 3DSG of the environment and the natural-language goal to an LLM, acting as a planner.

SayPlan [13] performs task planning on 3DSGs in a purely autoregressive manner, without an explicit symbolic planner. Because no official code was released, we reimplemented the method following the prompts and settings described in the paper in the same way as Liu et al. [24].

DELTA [24] is the state-of-the-art system for task planning on 3DSGs. It combines LLMs with PDDL planning to decompose the original task into feasible sub-goals for efficient problem solving. We use the officially released codebase, integrated into our framework.

B. Dataset

We adopt the 3DSG dataset of Armeni et al. [1], providing a collection of 3DSGs extracted from multi-room environments [8]. For comparability we follow the same task domains and scenarios as DELTA [24]: environments *Allensville, Parole, Shelbiana, Kemblesville* and tasks *Laundry (LA), PC Assembly (PC), Dining Table Setup (DS), House Cleaning (HC), Home Office Setup (OS)*[1]. We augment the original 3DSG with objects relevant to the task. Most tasks are solvable with the added objects. To stress adaptation, we introduce an additional set of problem instances that are intrinsically impossible, because some necessary objects are removed in the augmentation process, while semantically related substitutes or weaker variants are made available. This set comprises tasks that are not immediately satisfiable, due to unavailability of key objects, therefore requiring *shifts* or *relaxations* of the original goal. To assess the scalability of this approach, we introduce a *General (GN)* benchmark covering six additional environments (*Klickitat, Lakeville, Leonardo*, *Lindenwood, Markleeville, Marstons*), featuring generic pick-and-place tasks such as "*move all the apples from the kitchen to the fruit bowl in the dining room*". In total, our benchmark spans 141 tasks across 10 environments, compared to DELTA's 15 tasks in 4 environments. The official DELTA implementation augments 3DSGs with post-hoc attributes and states absent from the original dataset. For a fair comparison without extra information, we evaluate strictly on the unmodified dataset.

C. Metrics

The experimental evaluation focuses mainly on the *Success Rate* (SR) of the planning and grounding process, as the ratio between the successful tasks and the total number of tasks.

[1]For task definitions, see [24].

TABLE I

Comparison of baselines and our approach, w/ and w/o domain generation and grounding (Gr) in the environment.

			SR (%) Grounding + Planning	SR (%) Planning only	Avg. Planning time (s)	Avg. Plan Length
Domain generation	*CM*	*Gr*	66.94	88.15	38.68	15.72
	w/o Relaxation (ours)	*w/o Gr*	-	-	-	-
	CM	***Gr***	**91.73**	95.52	19.0	17.35
	w/ Relaxation (ours)	*w/o Gr*	-	**99.02**	24.24	17.61
	DELTA [24]	*Gr*	39.28	65.76	**0.03**	23.52
		w/o Gr	-	49.14	**0.03**	17.12
w/o Domain Generation	*CM* (ours)	*Gr*	**91.54**	**97.22**	9.94	17.81
	DELTA [24]	*Gr*	13.89	18.86	**0.02**	11.74
	SayPlan [13]	*w/o Gr*	-	46.0	28.15	19.6
	LLMAsPlanner	*w/o Gr*	-	71.2	8.84	21.6

A task is considered solved if both the planning and the grounding in the 3DSG are successful. Table I also shows the average SR of the planning step alone (considering only successful planning) and the *Average Planning Time*.

D. Experiments

We used *GPT-4o* for all experiments, with temperature set to 0 to minimize randomness in the output. We validate the symbolic syntax of both the outputs of DELTA and *ContextMatters*using VAL [25], and then with a grounding check. To this aim, the plan is virtually executed using PDDLGym [26], ensuring that no object label or location hallucinations are introduced by the LLM. The end-to-end nature of methods like SayPlan and LLMAsPlanner prevent the use of formal validation tools. As a result, they cannot offer strong guarantees of plan correctness and feasibility, a key limitation our symbolic framework is designed to overcome. SayPlan and LLMAsPlanner cannot benefit from the same grounding check due to the lack of a symbolic planning domain.

E. Discussion

Table I shows that *ContextMatters* achieves the best results on all tasks, both with and without domain generation. The LLMAsPlanner and SayPlan baselines show how LLMs are not directly applicable to the generation of groundable symbolic plans: in a more structured approach, the LLM is used to model the planning domain, then used to compute the plan. Our main architecture (in bold) features both domain generation and grounding checks, providing feedback to relax and refine the PDDL problem, for better feasibility of the computed plan. Experiments show that relaxations, intended as any combination of goal shifting and semantic relaxation, improve the SR compared to open-loop architectures. The first row of Table I shows that both the Planning and Grounding SR of *ContextMatters* decrease without goal relaxations. Following Algorithm 1, more relaxations translate to more allowed iterations of the outer loop, progressively shifting the planning domain and adapting it to the available objects in the 3DSG, improving the chances of correcting any syntax or modeling issues in the PDDL domain formulation. While DELTA computes the SR only on successfully generated plans, assuming

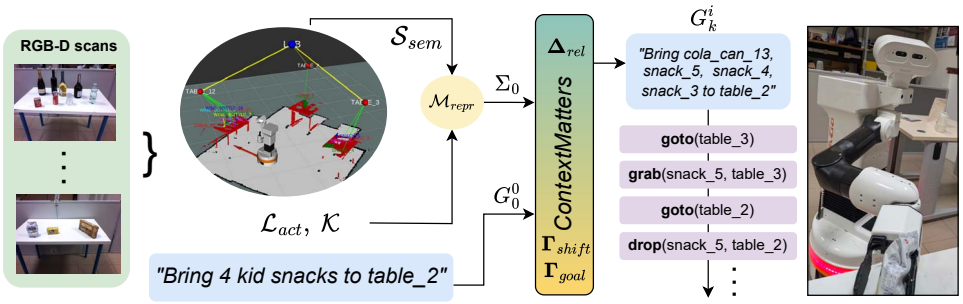

Fig. 3. Pipeline employed to deploy *ContextMatters* on a real robot. From perception we build a 3DSG of the environment, fed into our pipeline with the initial goal. We obtain the feasible goal in output, which is then translated into a sequence of groundable actions, executed by our robot.

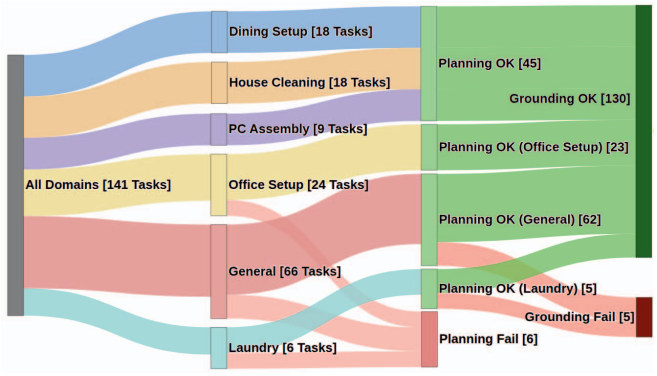

Fig. 4. Results of *ContextMatters* across our six-domain dataset, evaluated without domain generation and with grounding on the scene graph.

TABLE II

AVERAGE PERFORMANCE ON THE GENERAL (GN), HOUSE CLEANING (HC), PC ASSEMBLY (PC), OFFICE SETUP (OS), LAUNDRY (LA), AND DINING SETUP (DS) TASKS.

	SR (%)	Plan Length	Planning Time (s)	Expanded Nodes	Inference Time (s)
HC	100.0	17.72	16.93	3971.83	57.86
PC	100.0	23.22	9.5	832689.56	33.18
DS	100.0	27.22	53.71	1237733.28	89.12
GN	93.55	15.23	15.01	15441.76	54.9
LA	83.33	15.0	14.13	15039.0	169.78
OS	95.83	5.7	4.7	80.96	217.14

that the 3DSG is faithfully reproduced throughout the PDDL generation, we add a grounding check, verifying that plan actions reference existing domain objects in the 3DSG. In case of failure, grounding errors provide corrective feedback to the following iterations. While DELTA optimizes planning time by decomposing the planning problem in sub-problems, its performance in the grounding step is notably worse due to the lack of a self-supervised PDDL refinement loop, which proved essential in *ContextMatters* for correcting PDDL syntax errors and hallucinations.The Sankey diagram in Fig. 4 shows the performance across all 141 tasks, highlighting overall end-to-end success rates of *ContextMatters* while also clearly isolating failure modes.Table II shows the performances of the main architecture on each task, ordered by difficulty, with several statistics, including time spent for LLM inference over a problem and the corresponding Planning SR. Easier tasks, requiring simple pick-move-place actions, consist in collecting objects scattered in the 3DSG, therefore the planner tends to produce longer plans by searching in a broader space, as shown by the longer planning time and the higher number of expanded nodes along the search tree. The two most difficult tasks, Office Setup and Laundry, require better domain modeling by the LLM, and better adaptation, causing longer inference times, while the required planning effort is lower.

F. Real robot setup

We tested the architecture on a real TIAGo robot, serving food within a real environment [2]. The 3DSG was generated using the EMPOWER architecture [19], using RGB-D images to reproject semantic information from the camera into the 3D environment. Panoptic masks on detected point clouds are used to create a 3DSG following the dataset structure of [1] (Fig. 3). We employed *ContextMatters* without domain generation, with the scene graph grounding check, enabling planning in the real environment with plan feasibility constraints. The domain description was modeled to map the high-level PDDL actions to the physical capabilities of the TIAGo robot. (*e.g.*,, manipulating objects with the gripper). The robot arm is controlled with the *MoveIt!* motion planning framework [27]. The experiment was conducted with the natural language task: *"Bring 4 kid snacks to* `table_2`*"*, appropriately relaxed to: *"Bring* `cola_can_13`*,* `snack_5`*,* `snack_4`*,* `snack_3` *to* `table_2`*"*. Given that only three snacks were available in the environment, goal shifting replaced one snack with a cola can (as it is commonsense that wine bottles cannot be served to kids). The resulting plan is correctly executable in the real environment, successfully moving the required objects from `table_3` to `table_2`. Full execution is described in the supplementary material.

V. CONCLUSION

In this work, we presented *ContextMatters* a novel framework addressing a fundamental challenge in embodied AI planning: the gap between user intent and environmental constraints. Our approach introduces a bidimensional relaxation mechanism that systematically searches across both functional equivalence and feasibility dimensions,

[2]https://pal-robotics.com/robot/tiago/

enabling robots to adapt goals to their operational context while preserving task semantics.

Our key contribution lies in the formalization of contextual goal relaxation through the integration of situational shift (Γ_{shift}) and relaxation (Δ_{rel}) operators, enabling principled adaptation when exact goal satisfaction proves impossible. By combining the commonsense reasoning capabilities of Large Language Models with the formal guarantees of classical PDDL planning, *ContextMatters* demonstrates that seemingly infeasible tasks can often be transformed into executable plans through intelligent goal adaptation.

Quantitative results show that our approach achieves a substantial improvement over state-of-the-art methods, and the successful deployment on a TIAGo robot further validates the practical applicability of our framework in real-world scenarios, where perfect environmental conditions rarely align with idealized task specifications.

We believe that the ability to intelligently relax and adapt goals represents a crucial step toward truly robust embodied AI systems. As we continue to deploy robots in unstructured, real-world environments, the capacity to reason about what can be achieved given the context, rather than failing on unmet preconditions, will prove essential for practical autonomy.

ACKNOWLEDGEMENTS

This work has been carried out while Emanuele Musumeci, Michele Brienza and Francesco Argenziano were enrolled in the Italian National Doctorate on Artificial Intelligence run by Sapienza University of Rome. Michele Brienza is founded by the European Union - Next Generation EU, Mission I.4.1 Borse PNRR Pubblica Amministrazione (Missione 4) Component 1 CUP B53C23003540006. This work has been partially supported by PNRR MUR project PE0000013-FAIR.

REFERENCES

[1] I. Armeni, Z.-Y. He, J. Gwak, A. R. Zamir, M. Fischer, J. Malik, and S. Savarese, "3d scene graph: A structure for unified semantics, 3d space, and camera," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 5664–5673.

[2] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[3] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altenschmidt, S. Altman, S. Anadkat *et al.*, "Gpt-4 technical report," *arXiv preprint arXiv:2303.08774*, 2023.

[4] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozi'ere, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[5] J. Li, M. Zhang, N. Li, D. Weyns, Z. Jin, and K. Tei, "Generative ai for self-adaptive systems: State of the art and research roadmap," *ACM Transactions on Autonomous and Adaptive Systems*, vol. 19, no. 3, pp. 1–60, 2024.

[6] C. Aeronautiques, A. Howe, C. Knoblock, I. D. McDermott, A. Ram, M. Veloso, D. Weld, D. W. Sri, A. Barrett, D. Christianson *et al.*, "Pddl— the planning domain definition language," *Technical Report, Tech. Rep.*, 1998.

[7] J. Wald, H. Dhamo, N. Navab, and F. Tombari, "Learning 3d semantic scene graphs from 3d indoor reconstructions," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020.

[8] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese, "Gibson env: real-world perception for embodied agents," in *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*. IEEE, 2018.

[9] N. Hughes, Y. Chang, and L. Carlone, "Hydra: A real-time spatial perception system for 3d scene graph construction and optimization," 2022.

[10] J. Strader, N. Hughes, W. Chen, A. Speranzon, and L. Carlone, "Indoor and outdoor 3d scene graph generation via language-enabled spatial ontologies," 2024.

[11] A. Werby, C. Huang, M. Büchner, A. Valada, and W. Burgard, "Hierarchical open-vocabulary 3d scene graphs for language-grounded robot navigation," in *Robotics: Science and Systems XX*, ser. RSS2024. Robotics: Science and Systems Foundation, Jul. 2024.

[12] 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," *arXiv preprint arXiv:2309.16650*, 2023.

[13] K. Rana, J. Haviland, S. Garg, J. Abou-Chakra, I. Reid, and N. Suenderhauf, "Sayplan: Grounding large language models using 3d scene graphs for scalable task planning," *arXiv preprint arXiv:2307.06135*, 2023.

[14] C. Agia, K. M. Jatavallabhula, M. Khodeir, O. Miksik, V. Vineet, M. Mukadam, L. Paull, and F. Shkurti, "Taskography: Evaluating robot task planning over large 3d scene graphs," 2022.

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

[16] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch, "Language models as zero-shot planners: Extracting actionable knowledge for embodied agents," in *International Conference on Machine Learning*. PMLR, 2022, pp. 9118–9147.

[17] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "Llm-planner: Few-shot grounded planning for embodied agents with large language models," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 2998–3009.

[18] V. S. Dorbala, J. F. Mullen Jr, and D. Manocha, "Can an embodied agent find your" cat-shaped mug"? llm-guided exploration for zero-shot object navigation," *arXiv preprint arXiv:2303.03480*, 2023.

[19] F. Argenziano, M. Brienza, V. Suriani, D. Nardi, and D. D. Bloisi, "Empower: Embodied multi-role open-vocabulary planning with online grounding and execution," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2024, pp. 12 040–12 047.

[20] J. Yang, X. Chen, S. Qian, N. Madaan, M. Iyengar, D. F. Fouhey, and J. Chai, "Llm-grounder: Open-vocabulary 3d visual grounding with large language model as an agent," 2023.

[21] F. Joublin, A. Ceravola, P. Smirnov, F. Ocker, J. Deigmoeller, A. Belardinelli, C. Wang, S. Hasler, D. Tanneberg, and M. Gienger, "Copal: corrective planning of robot actions with large language models," in *2024 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2024, pp. 8664–8670.

[22] Y. Ding, X. Zhang, S. Amiri, N. Cao, H. Yang, A. Kaminski, C. Esselink, and S. Zhang, "Integrating action knowledge and llms for task planning and situation handling in open worlds," *Autonomous Robots*, vol. 47, no. 8, pp. 981–997, 2023.

[23] B. Liu, Y. Jiang, X. Zhang, Q. Liu, S. Zhang, J. Biswas, and P. Stone, "Llm+ p: Empowering large language models with optimal planning proficiency," *arXiv preprint arXiv:2304.11477*, 2023.

[24] Y. Liu, L. Palmieri, S. Koch, I. Georgievski, and M. Aiello, "Delta: Decomposed efficient long-term robot task planning using large language models," *arXiv preprint arXiv:2404.03275*, 2024.

[25] R. Howey, D. Long, and M. Fox, "Validating plans with exogenous events," in *Proceedings of the 23rd Workshop of the UK Planning and Scheduling Special Interest Group (PlanSIG 2004)*, 2004, pp. 78–87.

[26] T. Silver and R. Chitnis, "Pddlgym: Gym environments from pddl problems," *arXiv preprint arXiv:2002.06432*, 2020.

[27] D. Coleman, I. Sucan, S. Chitta, and N. Correll, "Reducing the barrier to entry of complex robotic software: a moveit! case study," *arXiv preprint arXiv:1404.3785*, 2014.