Plan Verification for LLM-Based Embodied Task Completion Agents

Ananth Hariharan, Vardhan Dongre, Dilek Hakkani-Tür, Gokhan Tur
University of Illinois Urbana-Champaign
{ahari8, vdongre2, dilek, gokhan}@illinois.edu

Abstract

Large language model (LLM) based task plans and corresponding human demonstrations for embodied AI may be noisy, with unnecessary actions, redundant navigation, and logical errors that reduce policy quality. We propose an iterative verification framework in which a Judge LLM critiques action sequences and a Planner LLM applies the revisions, yielding progressively cleaner and more spatially coherent trajectories. Unlike rule-based approaches, our method relies on natural language prompting, enabling broad generalization across error types including irrelevant actions, contradictions, and missing steps. On a set of manually annotated actions from the TEACh embodied AI dataset, our framework achieves up to 90% recall and 100% precision across four state-of-the-art LLMs (GPT o4-mini, DeepSeek-R1, Gemini 2.5, LLaMA 4 Scout). The refinement loop converges quickly, with 96.5% of sequences requiring at most three iterations, while improving both temporal efficiency and spatial action organization. Crucially, the method preserves human error-recovery patterns rather than collapsing them, supporting future work on robust corrective behavior. By establishing plan verification as a reliable LLM capability for spatial planning and action refinement, we provide a scalable path to higher-quality training data for imitation learning in embodied AI. [1]

1 Introduction

Recent advances in large language models (LLMs) have enabled sophisticated multi-agent systems for handling task plans in embodied AI. Embodied agents deployed in real-world environments are often tasked with executing extended sequences of actions in pursuit of high-level goals. For such agents to be effective, their plans must be accurate in terms of task completion, efficient in their use of time and actions, and safe in avoiding undesirable or unintended consequences. With the advent of LLMs, there has been growing interest in using these systems to generate and reason about structured plans from natural language input. Recent work has demonstrated that LLMs can decompose high-level goals into executable sub-goals and, in multi-agent setups, evaluate and revise each other's outputs through structured dialogue ([1], [2]).

Despite this progress, LLM-generated plans have been shown to be not perfect [3, among others], and need to be verified. Furthermore, the quality of the datasets used to train planning for embodied agents remains a significant bottleneck. Many widely used corpora, such as TEACh ([4]), a dataset of household interactions collected via human operation, contain a large number of suboptimal behaviors ([5]). These include, for example, picking up irrelevant or unnecessary items, leaving goals unaccomplished due to erroneous instruction, or toggling appliances that were never activated.

---

[1]Code and models: https://github.com/AnanthHariharan/Task-Agents

Planning Agent

| Initial Dataset | Plan Creation | Goal Annotation | Final Dataset |

Initial Dataset
```
"action": {"action_id":
103, "action_name":
"PickUp X",
"action_params":
["Potato"]
```

Plan Creation
```
Driver.
PickUp('Potato'),
Driver.PutAonB('Mug',
'CoffeeMachine')
```

Goal Annotation
```
# GOAL: Prepare
coffee in a mug
```

Final Dataset
```
Driver.Pickup('Mug')
Driver.Move(0.25)
Driver.PutAonB('Mug',
'CoffeeMachine')
Driver.ToggleOn('Coffe
eMachine')
```

Iteration until zero changes

NL Action Annotation
```
Driver.PickUp('Potato')
// Driver picks up the potato
Driver.PutAonB('Mug',
'CoffeeMachine')
// Driver places mug on coffee
machine
```

Actions analyzed with "GOAL" statement

Redundancy/Missing Tagging
```
# REMOVE: Unnecessary to pick up
potato to fill coffee pot
# MISSING: Coffee machine not
turned on
```
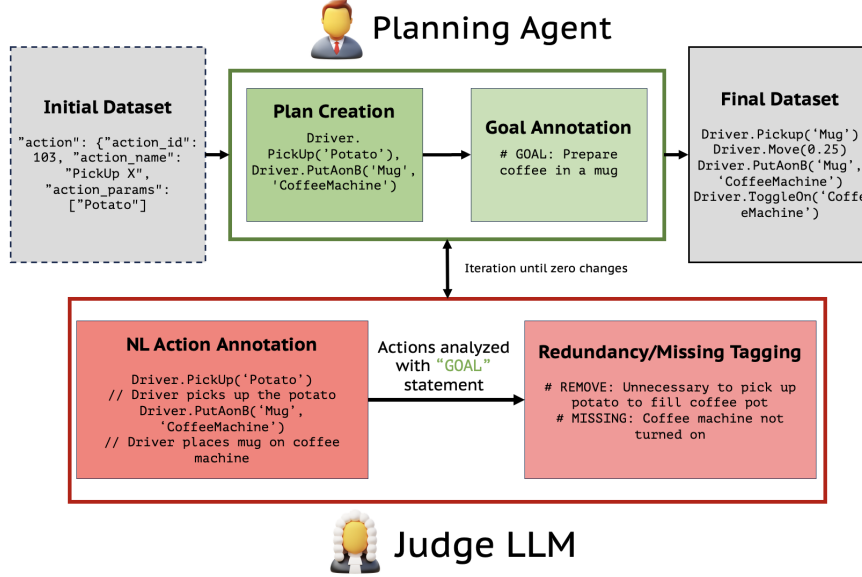
Judge LLM

Figure 1. Diagram of Planning Agent and Judge LLM Interaction Process for Plan Verification

Figure 1: Diagram of Planning Agent and Judge LLM Interaction Process for Plan Verification

While such issues result in unnecessarily lengthy plans and introduce noise in learning signals during imitation or reinforcement learning, they also present an underexplored opportunity: human demonstrations naturally contain error-recovery sequences that showcase how humans recognize and correct mistakes in real-time. Our verification framework, by explicitly identifying these erroneous actions, not only enables cleaner training data through removal but also facilitates future research into learning from human error recovery patterns, potentially yielding agents that exhibit human-like resilience and adaptability when facing unexpected situations or their own execution errors.

To address these limitations, we propose a general verification framework based on a two-agent protocol. In our approach, a Planning Agent first generates a candidate plan for a given goal. This plan is then passed to a second agent – a Judge powered by an LLM – which analyzes the sequence step by step and flags actions that appear redundant, irrelevant, contradictory, or otherwise unjustified. Each flagged step is accompanied by a natural language explanation to support interpretability. The Planning Agent subsequently revises its plan in response to this feedback, and the process repeats until the Judge LLM returns no further objections or a predefined iteration limit is reached. This end-to-end sequence is described in Figure 1.

This approach is fully language-based, model-agnostic, and does not rely on any handcrafted heuristics or domain-specific rule sets. Instead, it leverages the reasoning capabilities of pretrained LLMs via zero-shot prompting, using a concise set of common-sense criteria embedded in natural language. The Judge LLM operates independently of any environment simulator or visual input, making the method simple to integrate into a wide range of embodied AI pipelines.

Our empirical evaluation focuses on the TEACh dataset, where we analyze 100 episodes spanning 15 high-level household tasks. We evaluate four LLMs in the Judge and Planning Agent role and report their effectiveness in identifying plan defects, measuring both precision and recall against a manually annotated ground truth. Our findings reveal a clear trade-off between conservative high-precision judges and more aggressive high-recall ones, with each offering complementary benefits for downstream applications.

Our contributions are:

1. **Framework:** We introduce a general-purpose language-based plan verification framework that incorporates iterative critique and revision into embodied planning workflows, represented in Figure 1.
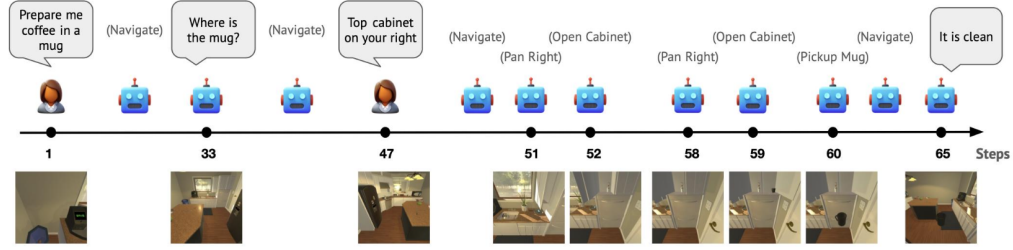
Figure 2: Diagram of Sample Workflow in TEACh Dataset

2. **NL Prompt-Based Judge LLM:** We demonstrate that simple natural language-based rules, encoded in prompts, are sufficient to guide LLM-based judges in identifying and explaining suboptimal actions.

3. **Empirical Study:** We benchmark multiple Judge LLMs on the TEACh dataset and provide detailed analysis of their respective strengths, weaknesses, and trade-offs. The structure of these workflows is illustrated in Figure 2.

This work lays the groundwork for scalable, language-driven refinement of human-authored task plans in embodied settings, with potential downstream benefits for both imitation learning and reinforcement learning paradigms.

2 Related Work

Our work on plan verification intersects with a broad effort to integrate LLMs into complex, multi-step reasoning pipelines. We situate our framework by synthesizing research across three major themes: the challenges and advancements in using LLMs as direct plan generators; the emergence of LLMs as verifiers and refiners in iterative feedback loops; and the broader context of the LLM-as-a-Judge ecosystem, including its benchmarks and known vulnerabilities.

2.1 LLMs as Plan Generators

In the earlier work, Gella et al. have demonstrated the dialogue act prediction based approach using the TEACh dataset [6]. This was a classification based approach trying to predict the next dialogue act given the action history and environment. They have expanded this work so as to use an LLM based planning approach in [3] to generate an entire action sequence from a high-level goal. Another work by Huang et al. [7] demonstrated that while large pre-trained models contain sufficient world knowledge to produce semantically plausible plans, these plans are often not executable because they fail to map onto an agent's specific, admissible actions. This fundamental gap between plausible text and executable actions highlights the core challenge of grounding.

Subsequent research has focused on closing this gap by providing LLMs with stronger environmental context. For instance, PROGPROMPT [8] structures the prompt as a Python program, providing the LLM with available actions as import statements and objects as a list, thereby constraining the generated output to be more syntactically correct and environmentally aware. Similarly, LLM-PLANNER [9] couples few-shot prompting with online simulator feedback, appending error traces to the prompt to enable replanning when a step fails. Other methods, like TAPA [10], inject explicit world state, such as an inventory table, directly into the prompt to reduce object hallucination. Our work is orthogonal to these plan generation methods; instead of improving the initial output, we focus on creating a robust post-hoc verification process to clean noisy, human-authored trajectories, ensuring that any downstream planner inherits a leaner and less ambiguous set of demonstrations.

2.2 Verification and Refinement

Given the difficulties in direct plan generation, a powerful alternative paradigm has emerged: using LLMs to critique, verify, and refine plans within an iterative loop. This approach often decomposes

the problem into distinct agentic roles. The Reflexion framework, for example, splits an agent into an Actor, Evaluator, and Self-Reflection module, using verbal reinforcement to iteratively improve its policy without gradient updates [1]. Our Judge-Planner framework externalizes this critique process into an independent, two-agent system, which facilitates explicit precision-recall accounting and allows for modular substitution of the Judge LLM. This multi-agent perspective is echoed in broader frameworks like AUTOGEN, which coordinates specialized conversable agents [2], and COELA, which pairs a "Commander" LLM with domain-specific modules for robot teams [11].

More directly, recent work has built systems specifically for pre-execution plan verification. VerifyLLM [12] proposes a verification architecture that analyzes action sequences to identify logical inconsistencies, missing prerequisites, and redundancies, using Linear Temporal Logic (LTL) as a formal intermediate representation to guide the LLM's analysis. The principle of using an LLM to verify an intermediate output is not limited to robotics; LLatrieval [13] employs an LLM to iteratively verify and refine the documents retrieved by a RAG system, demonstrating the general power of this verification-and-update pattern. Our framework contributes to this line of work by demonstrating that effective, high-recall verification can be achieved using zero-shot natural language critique alone, without requiring formal methods like LTL.

2.3 "LLM-as-a-Judge" Ecosystem

Casting an LLM in a verification role makes it an instance of an "LLM-as-a-Judge," a rapidly growing field with its own set of tools and challenges. A systematic analysis by Li et al. [14] formally evaluates the distinct contributions of LLMs as solvers, verifiers, and heuristic functions, concluding that LLMs are often more effective at providing comparative feedback than at generating correct solutions from scratch – a finding that strongly motivates our verification-centric approach.

However, the reliability of LLM judges is not guaranteed. Research into their adversarial robustness has shown that simple, universal triggers can inflate judge scores, highlighting the fragility of naive evaluation [15]. To promote standardized evaluation, public benchmarks such as MT-BENCH and the CHATBOT ARENA have been developed to provide open, crowd-validated tests where models like GPT-4 have achieved human-level agreement [16]. Our work adopts their strict evaluation protocols. Furthermore, systematic surveys have begun to chart the field, offering taxonomies of judgment tasks and techniques for bias mitigation [17, 18], which inform the design of our evaluation suite. Finally, within the embodied domain, benchmarks like PARTNR provide large-scale, multi-agent tasks with built-in verification checkpoints [19]. We instead target the TEACh dataset for its organic, noisy human demonstrations, which better approximate the data our framework is designed to clean.

3 Methodology

Our objective is to automatically refine *human-authored* embodied task trajectories by identifying and correcting actions that are irrelevant, redundant, contradictory, or missing relative to the stated goal. We cast this process as an interaction between two language model-based agents: a **Judge LLM**, which critiques a proposed plan, and a **Planning Agent**, which applies those critiques to produce an updated plan. This modular design yields a transparent, fully language-driven verification loop that can be applied to existing datasets without retraining.

3.1 Formal Problem Definition

Let Π denote the space of action sequences (plans). A plan $\pi \in \Pi$ is a finite sequence

$$\pi = (a_1, a_2, \ldots, a_T),$$

where each a_t is an atomic manipulation such as `PickUp`, `ToggleOn`, or `Place`. Not all actions in π are correct. We consider an *error set* $\mathcal{E}(\pi)$ containing the positions of erroneous actions in π. Errors arise in three forms:

1. **Redundant actions** (unnecessary repetitions or canceling operations).
2. **Contradictory actions** (steps that oppose earlier actions or the task goal; treated as `REMOVE`).
3. **Missing actions** (omissions required for successful task completion).

Objective. Given a natural language goal g and an initial plan $\pi^{(0)}$, our task is to produce a refined plan π^* that achieves g while minimising length:

$$\pi^* = \arg\min_{\tilde{\pi}} |\tilde{\pi}| \quad \text{s.t.} \quad \tilde{\pi} \text{ achieves } g.$$

This formulation permits insertions when necessary, but penalises unnecessary actions.

Verification Operator. The Judge implements a critique function

$$J : (g, \pi) \mapsto \mathcal{C} = \{(i, \text{type}, \text{reason})\},$$

where type $\in \{\texttt{REMOVE}, \texttt{MISSING}\}$ and each critique specifies an action index i, a correction type, and a rationale. The Planner applies these critiques deterministically:

$$P : (\pi, \mathcal{C}) \mapsto \pi'.$$

We define the verification operator as the composition

$$V = P \circ J,$$

so that $\pi' = V(g, \pi)$. We assume V is *conservative*, i.e.

$$\mathbb{E}[E(V(\pi))] \leq E(\pi),$$

where $E(\pi) = |\mathcal{E}(\pi)|$ is the error count.

Iterative Refinement. Verification is applied iteratively. At iteration k, the Judge produces critiques $\mathcal{C}^{(k)} = J(g, \pi^{(k)})$, and the Planner incorporates them:

$$\pi^{(k+1)} = P(\pi^{(k)}, \mathcal{C}^{(k)}).$$

The error counts form a non-increasing sequence $\{E^{(k)}\}$. Under the conservative assumption, there exists $\delta > 0$ such that

$$\mathbb{E}[E^{(k+1)}] \leq (1 - \delta)\,\mathbb{E}[E^{(k)}],$$

suggesting geometric convergence to a fixed point $E^* \geq 0$. Empirically, convergence occurs within three iterations in 96.5% of cases.

3.2 Evaluation Metrics

We evaluate the Judge's ability to detect errors against human annotations.

- **Recall** $= \frac{\text{TP}}{\text{TP+FN}}$: fraction of erroneous actions flagged.
- **Precision** $= \frac{\text{TP}}{\text{TP+FP}}$: fraction of flagged actions that were truly erroneous.
- **F$_1$ score**: harmonic mean of precision and recall.

Here positives are erroneous actions (REMOVE or MISSING); contradictory errors are subsumed under REMOVE. Task-level outcomes (e.g., plan success rate, average length reduction) are reported separately.

3.3 Iterative Verification Algorithm

Algorithm 1 instantiates the operator $V = P \circ J$ as an iterative refinement loop.

4 Results

We report results from both the single-pass (zero-shot) and multi-pass (iterative) evaluation settings. These results are obtained by manual annotations that assess the final Planning Agent output based on the action sequence accurately achieving the goal set by the Commander of each TEACh action sequence.

Algorithm 1 Iterative Plan Verification (Judge–Planner Composition)

Require: Goal g, initial plan $A = (a_1, \ldots, a_n)$, Judge LLM J, Planner LLM P
Ensure: Refined plan A'
   $A' \leftarrow A$
   for $i = 1$ to 5 **do**   ▷ Maximum refinement rounds
      $critiques \leftarrow J.\text{evaluate}(g, A')$
      if $critiques = \emptyset$ **then**
         return A' ▷ No errors detected
      end if
      $A' \leftarrow P.\text{apply\_critiques}(A', critiques)$
      $A' \leftarrow \text{reindex}(A')$
   end for
   return A' ▷ Return last refined plan if critiques persist

4.1 Static (Zero-Shot) Verification Performance

Table 1 summarizes the recall and precision of each Judge LLM when applied directly to raw TEACh plans. GPT o4-mini achieved the highest overall recall (80%) while maintaining a strong precision of 93%. DeepSeek-R1 exhibited perfect precision (100%) but with a lower recall of 68%, indicating a more conservative judging style. Gemini 2.5 and LLaMA 4 Scout provided moderately strong recall (74%) with slightly reduced precision (90% and 85%, respectively). The rule-based baseline underperformed across both metrics, as shown in Table 1.

Judge LLM	Recall	Precision
GPT o4-mini	**80%**	93%
DeepSeek-R1	68%	**100%**
Gemini 2.5	74%	90%
LLaMA 4 Scout	74%	85%
Rule-based	22%	71%

Table 1: Single-pass plan verification performance.

4.2 Iterative Critique-and-Revise Performance

We now examine how plan quality improves over multiple critique–and-revise rounds. Each Planning Agent LLM proposes a revised plan after receiving feedback from the Judge LLM, and this loop continues until the Judge LLM raises no new issues.

As shown in Table 2, iterative refinement boosts recall by 5–10% on average across all Judge–Planner combinations. For instance, GPT o4-mini's recall improves from 80% (static) to 88–90% when paired with itself or other Planner LLMs. Similarly, Gemini 2.5 sees recall rise to 89% with multiple planner pairings. Precision generally remains stable or improves slightly.

To understand the efficiency of the iterative refinement process, we analyzed the convergence behavior across all each of the individual steps over all actions cumulatively judged and planned by the LLM agents. Figure 3 illustrates the cumulative percentage of sequences that reach their final state after each iteration.

Our analysis reveals:

- **Iteration 1**: 62% of sequences require no further modifications
- **Iteration 2**: 89% cumulative convergence (+27%)
- **Iteration 3**: 96.5% cumulative convergence (+7.5%)
- **Iterations 4-5**: Only 3.5% of sequences benefit from additional rounds

This rapid convergence suggests that most plan defects are straightforward and can be identified in a single pass, while a smaller subset requires iterative reasoning to resolve complex interdependencies between actions.
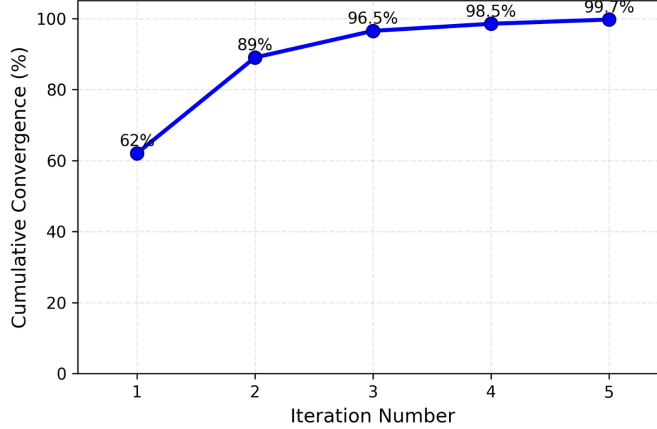
Figure 3: Cumulative convergence of action sequences across iterations. Most sequences (62%) are corrected after the first iteration, with near-complete convergence (96.5%) by iteration 3.

Judge LLM	Planner LLM – Recall (%) / Precision (%) / F-score			
	GPT o4-mini	DeepSeek-R1	Gemini 2.5	LLaMA 4 Scout
GPT o4-mini	88 / 90 / 89.0	**90 / 80 / 84.7**	85 / 91 / 87.8	89 / 87 / 87.9
DeepSeek-R1	65 / 99 / 78.5	**68 / 100 / 80.9**	62 / 100 / 76.5	66 / 98 / 78.9
Gemini 2.5	84 / 98 / 90.7	86 / 97 / 91.2	**89 / 99 / 93.9**	89 / 96 / 92.2
LLaMA 4 Scout	76 / 92 / 83.5	81 / 90 / 85.3	79 / 93 / 85.9	75 / 89 / 81.6

Table 2: Iterative recall and precision (%): Recall is the percentage of ground-truth RE-MOVE/MISSING actions correctly flagged; precision is the percentage of flagged actions that were genuinely erroneous

4.3 Qualitative Analysis and Error Patterns

Our manual analysis of Judge LLM outputs reveals systematic patterns in both successful corrections and failure modes. We categorize these findings by error type and precision/recall failures.

4.3.1 Model Behavior

Our analysis reveals distinct behavioral patterns among the four Judge LLMs evaluated. **DeepSeek-R1** emerges as the most conservative judge, achieving near-perfect precision (98-100%) across all planner pairings but at the cost of significantly lower recall (62-68%). This conservative approach means DeepSeek-R1 only flags actions when absolutely certain, resulting in many problematic actions passing through undetected. While this minimizes false positives, it fails to identify approximately one-third of the errors in human-authored plans, limiting its effectiveness for comprehensive plan refinement.

In contrast, **GPT o4-mini** demonstrates a more balanced but aggressive approach, achieving the highest recall rates (85-90%) while maintaining respectable precision (80-91%). This model excels at catching subtle errors that other judges miss, particularly when paired with DeepSeek-R1 as the planner (90% recall). However, its eagerness to flag potential issues occasionally leads to over-correction, with precision dropping to 80% in some configurations. This trade-off makes GPT o4-mini ideal for scenarios where catching all errors is paramount, even if some valid actions are incorrectly flagged.

Gemini 2.5 presents the most well-rounded performance profile, consistently achieving high F1-scores (90.7-93.9) that indicate excellent balance between precision and recall. With recall rates of 84-89% and precision consistently above 96%, Gemini 2.5 demonstrates sophisticated judgment capabilities that avoid both the over-conservatism of DeepSeek-R1 and the occasional over-eagerness of GPT o4-mini. Notably, Gemini 2.5 performs best when paired with itself or LLaMA 4 Scout as planners, suggesting strong internal consistency in its evaluation criteria.

7

LLaMA 4 Scout occupies a middle ground, with moderate recall (75-81%) and good precision (89-93%). While it doesn't excel in any particular metric, its consistent performance across different planner pairings suggests robustness to varying input styles. LLaMA 4 Scout appears to focus on identifying the most obvious errors while being cautious about borderline cases, making it a reliable if unspectacular choice for plan verification tasks.

The interaction effects between judge-planner pairs also reveal interesting patterns. Models generally perform best when paired with themselves (as seen in the diagonal of our results table), suggesting that each model has internally consistent standards for what constitutes correct action sequences. However, some cross-model pairings yield surprising benefits: GPT o4-mini as judge with DeepSeek-R1 as planner achieves the highest recall (90%), while Gemini 2.5 consistently achieves the best overall performance regardless of planner choice, indicating its superior generalization capabilities.

4.3.2 Successful Corrections

Judge LLMs consistently identify and correct:

- **Premature toggles**: `REMOVE: Driver toggles off microwave before turning it on`
- **Irrelevant objects**: `REMOVE: Driver picks up RemoteControl - not needed for cooking task`
- **Incomplete sequences**: `MISSING: Task incomplete - bread sliced but sandwich not assembled`

4.3.3 Recall Failures (Missed Errors)

The Judge LLMs systematically miss certain error patterns, which are marked by the annotator's notes (AN) in the examples below:

Context-dependent redundancies. In sequences where objects are picked up early but used much later, judges fail to recognize the inefficiency:

```
// GOAL: Clean the bathroom
Driver.PickUp('Soap')      // Picked up early
Driver.Move(5.0)           // Multiple intervening actions
Driver.Turn(90)
Driver.PickUp('Sponge')
Driver.Place('Sink')       // Soap finally used here
// AN: Judge fails to flag early pickup as inefficient
```

4.3.4 Precision Failures (False Positives)

Judge LLMs occasionally over-correct valid sequences:

Multi-step preparations. Judges sometimes flag necessary preparatory actions as redundant:

```
// GOAL: Make coffee
Driver.PickUp('Mug')         // REMOVE (AN: Incorrectly flagged)
Driver.Place('Counter')
Driver.PickUp('CoffeeFilter')
Driver.Place('CoffeeMachine')
```

Object reuse patterns. When objects serve multiple purposes, judges may incorrectly remove secondary uses:

```
// GOAL: Set dinner table
Driver.PickUp('Plate')
Driver.Place('DiningTable')
Driver.PickUp('Plate')       // REMOVE (AN: Incorrectly flagged)
Driver.Place('DiningTable') // Actually placing second plate
```

These error patterns suggest that while current Judge LLMs excel at surface-level logical inconsistencies, they may struggle with long-range dependencies and context-sensitive reasoning.

5   Conclusion

This paper presents a general-purpose, language-driven framework for verifying human-authored embodied task plans via structured LLM-based critique. By casting plan verification as a modular dialogue between a Planning Agent and a Judge LLM, we enable scalable, interpretable, and model-agnostic refinement of noisy demonstration datasets without the need for task-specific heuristics or environment simulators. We evaluated the four LLMs GPT o4-mini, DeepSeek-R1, Gemini 2.5, and LLaMA 4 Scout as Judges on a curated slice of the TEACh dataset comprising 1,408 human-generated actions. Our findings reveal consistent advantages of natural language-based verification over rule-based baselines, with zero-shot prompting alone yielding up to 80% recall and 100% precision depending on the model. Moreover, we demonstrate that iterative critique-and-revise loops provide a consistent 5–10% boost in recall without sacrificing precision, underscoring the complementary power of multi-round plan inspection and LLM collaboration. Qualitative analysis confirms that Judge LLMs produce human-interpretable, step-level rationales that support transparent auditability – providing a more interpretable alternative. We also identify trade-offs between precision-focused and recall-focused Judge LLMs, suggesting hybrid ensembles or confidence-calibrated prompts as promising future directions.

These results establish plan verification as a distinct, language-level capability of LLMs, with strong implications for both dataset curation and agent performance in downstream learning tasks. Future work will extend this framework to vision-grounded settings, explore adversarial robustness of judge prompts, and test the framework's scalability on larger multi-agent corpora such as PARTNR [19].

6   Limitations

Our study introduces a robust verification framework utilizing natural language-based critique-and-revision loops to refine action plans within embodied task scenarios. However, several limitations must be acknowledged. Firstly, our evaluation is conducted exclusively on a subset of the TEACh dataset, covering 15 common household goals, and the generalizability of our findings to broader, more diverse embodied tasks and environments (e.g., industrial settings, healthcare, or complex outdoor environments) remains untested. Further, our results depend significantly on manual labeling of actions as necessary or unnecessary, and although careful and systematic, this process inherently introduces subjective judgment and potential inconsistencies, suggesting that future research should explore automated or semi-automated annotation methods to enhance consistency and efficiency.

The efficacy of our approach hinges on the quality and reliability of the Judge LLMs used as well, and the zero-shot prompting method, while effective, may be limited by the LLM's inherent biases, knowledge cutoff dates, and potential hallucinations or inaccuracies. These issues could potentially be mitigated through fine-tuning or hybrid approaches integrating external knowledge bases. Moreover, while our iterative critique-and-revise loop is computationally lightweight once the Judge LLM responses are cached, the initial computational overhead of generating Judge LLM responses, especially with large-scale datasets or more complex tasks, could be considerable, necessitating optimizations or more efficient prompting strategies for scalability. Finally, our method currently relies solely on natural language cues for identifying redundant or erroneous actions, and integrating stronger environmental grounding (e.g., visual object recognition or physical simulations) may enhance the accuracy and reliability of the verification process. Addressing these limitations through further research will be essential to fully realize the potential of natural-language verification approaches in diverse and complex real-world embodied tasks.

7   Acknowledgements

References

[1] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, 2023. URL `https://arxiv.org/abs/2303.11366`.

[2] Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. Autogen: Enabling next-gen llm applications via multi-agent conversation. 2023. URL `https://arxiv.org/abs/2308.08155`.

[3] Aishwarya Padmakumar, Mert Inan, Spandana Gella, Patrick Lange, and Dilek Hakkani-Tur. Multimodal embodied plan prediction augmented with synthetic embodied dialogue. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 6114–6131, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.374. URL `https://aclanthology.org/2023.emnlp-main.374/`.

[4] Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tur. Teach: Task-driven embodied agents that chat. 2021. URL `https://arxiv.org/abs/2110.00534`.

[5] So Yeon Min, Hao Zhu, Ruslan Salakhutdinov, and Yonatan Bisk. Don't copy the teacher: Data and model challenges in embodied dialogue. 2022. URL `https://arxiv.org/abs/2210.04443`.

[6] Spandana Gella, Aishwarya Padmakumar, Patrick Lange, and Dilek Hakkani-Tur. Dialog acts for task-driven embodied agents. 2022. URL `https://arxiv.org/abs/2209.12953`.

[7] Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. 2022. URL `https://arxiv.org/abs/2201.07207`.

[8] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models. 2022. URL `https://arxiv.org/abs/2209.11302`.

[9] Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M. Sadler, Wei-Lun Chao, and Yu Su. Llm-planner: Few-shot grounded planning for embodied agents with large language models, 2023. URL `https://arxiv.org/abs/2212.04088`.

[10] Zhenyu Wu, Ziwei Wang, Xiuwei Xu, Jiwen Lu, and Haibin Yan. Embodied task planning with large language models, 2023. URL `https://arxiv.org/abs/2307.01848`.

[11] Hongxin Zhang, Weihua Du, Jiaming Shan, Qinhong Zhou, Yilun Du, Joshua B. Tenenbaum, Tianmin Shu, and Chuang Gan. Building cooperative embodied agents modularly with large language models. 2024. URL `https://arxiv.org/abs/2307.02485`.

[12] Danil S. Grigorev, Alexey K. Kovalev, and Aleksandr I. Panov. Verifyllm: Llm-based pre-execution task plan verification for robots. 2025. URL `https://arxiv.org/abs/2507.05118`.

[13] Xiaonan Li, Changtai Zhu, Linyang Li, Zhangyue Yin, Tianxiang Sun, and Xipeng Qiu. Lla-trieval: Llm-verified retrieval for verifiable generation. 2024. URL `https://arxiv.org/abs/2311.07838`.

[14] Haoming Li, Zhaoliang Chen, Songyuan Liu, Yiming Lu, and Fei Liu. Systematic analysis of llm contributions to planning: Solver, verifier, heuristic. 2024. URL `https://arxiv.org/abs/2412.09666`.

[15] Vyas Raina, Adian Liusie, and Mark Gales. Is llm-as-a-judge robust? investigating universal adversarial attacks on zero-shot llm assessment. 2024. URL `https://arxiv.org/abs/2402.14016`.

[16] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023. URL `https://arxiv.org/abs/2306.05685`.

[17] Jiawei Gu, Xuhui Jiang, Zhichao Shi, Hexiang Tan, Xuehao Zhai, Chengjin Xu, Wei Li, Yinghan Shen, Shengjie Ma, Honghao Liu, Saizhuo Wang, Kun Zhang, Yuanzhuo Wang, Wen Gao, Lionel Ni, and Jian Guo. A survey on llm-as-a-judge, 2025. URL `https://arxiv.org/abs/2411.15594`.

[18] Dawei Li, Bohan Jiang, Liangjie Huang, Alimohammad Beigi, Chengshuai Zhao, Zhen Tan, Amrita Bhattacharjee, Yuxuan Jiang, Canyu Chen, Tianhao Wu, Kai Shu, Lu Cheng, and Huan Liu. From generation to judgment: Opportunities and challenges of llm-as-a-judge. 2025. URL `https://arxiv.org/abs/2411.16594`.

[19] Matthew Chang, Gunjan Chhablani, Alexander Clegg, Mikael Dallaire Cote, Ruta Desai, Michal Hlavac, Vladimir Karashchuk, Jacob Krantz, Roozbeh Mottaghi, Priyam Parashar, Siddharth Patki, Ishita Prasad, Xavier Puig, Akshara Rai, Ram Ramrakhya, Daniel Tran, Joanne Truong, John M. Turner, Eric Undersander, and Tsung-Yen Yang. Partnr: A benchmark for planning and reasoning in embodied multi-agent tasks. *arXiv preprint arXiv:2205.08866*, 2022. Work done at FAIR Meta. Alphabetical author order.

A   Judge LLM and Planning Agent Prompts

The following prompt was used to instruct the LLMs to evaluate and flag unnecessary or erroneous actions.

A.1   Judge LLM Prompt

```
You are a Judge Agent for embodied AI task planning. Your role is to provide
 thoughtful,natural language feedback on action sequences. You should:
1. Analyze each action's purpose and relevance to the goal
2. Explain your reasoning in clear, conversational language
3. Point out redundant or unnecessary actions with detailed explanations
4. Identify missing actions needed to complete the goal
5. Focus on being helpful and constructive in your feedback
Provide your feedback as natural language commentary, using #REMOVE and
#MISSING tags only when necessary. Prioritize clear explanations.
Please evaluate this action sequence for achieving the following goal:
GOAL: {goal}
Action Sequence: {actions_text}
Provide line-by-line analysis of each action. For each action, explain what
it does and whether it's necessary for the goal. Use this format:
ACTION: [copy the exact action]
ANNOTATION: [explain what this action does and whether it's needed for the goal.
If the action should be removed, include "#REMOVE: reason".
If it's good, just explain why.]
After analyzing all actions, if any steps are missing to complete the goal, add:
#MISSING: [describe what actions are needed]
Be thorough and conversational in your explanations. Focus on helping someone
understand why each action is or isn't necessary for achieving the goal.
Your line-by-line analysis:
```

A.2   Planning Agent Prompt

```
You are a Planning Agent for embodied AI tasks. Your role is to:
1. Analyze action sequences and identify their goals
2. Modify action sequences based on feedback from a Judge
3. Remove redundant actions and add missing actions as needed
```

4. Ensure action sequences are efficient and complete
Always preserve the original format and only make necessary changes.
Analyze the following action sequence and determine the overall
Context: {context}
Actions:{actions_text}
Provide a concise goal statement starting with "GOAL: "