

Article

Not peer-reviewed version

Agentic Workflows for Improving LLM Reasoning in Robotic Object-Centered Planning

[Jesus Moncada-Ramirez](#) , [Jose-Luis Matez-Bandera](#) , [Javier Gonzalez-Jimenez](#) , [Jose-Raul Ruiz-Sarmiento](#)

*

Posted Date: 3 January 2025

doi: 10.20944/preprints202501.0131.v1

Keywords: large language models; robotics; agentic workflows; reflection; semantic maps; object-centered planning



Preprints.org is a free multidisciplinary platform providing preprint service that is dedicated to making early versions of research outputs permanently available and citable. Preprints posted at Preprints.org appear in Web of Science, Crossref, Google Scholar, Scilit, Europe PMC.

Copyright: This open access article is published under a Creative Commons CC BY 4.0 license, which permit the free download, distribution, and reuse, provided that the author and preprint are cited in any reuse.

Article

Agentic Workflows for Improving LLM Reasoning in Robotic Object-Centered Planning

Jesus Moncada-Ramirez, Jose-Luis Matez-Bandera, Javier Gonzalez-Jimenez
and Jose-Raul Ruiz-Sarmiento *

Machine Perception and Intelligent Robotics Group (MAPIR-UMA), Malaga Institute for Mechatronics Engineering and Cyber-Physical Systems (IMECH.UMA), University of Malaga, Spain

* Correspondence: jotaraul@uma.es

Abstract: Large Language Models (LLMs) provide cognitive capabilities that enable robots to interpret and reason about their workspace, especially when paired with semantically rich representations like semantic maps. However, these models are prone to generating inaccurate or invented responses, known as hallucinations, that can produce an erratic robotic operation. This can be addressed by employing agentic workflows, structured processes that guide and refine the model's output to improve response quality. This work formally defines and qualitatively analyzes the impact of three agentic workflows (*LLM Ensemble*, *Self-Reflection*, and *Multi-Agent Reflection*) on enhancing the reasoning capabilities of an LLM guiding a robotic system to perform object-centered planning. In this context, the LLM is provided with a pre-built semantic map of the environment and a query, to which it must respond by determining the most relevant objects for the query. This response can be used in a multitude of downstream tasks. Extensive experiments were carried out employing state-of-the-art LLMs and semantic maps generated from the widely-used datasets ScanNet and SceneNN. Results show that agentic workflows significantly enhance object retrieval performance, especially in scenarios requiring complex reasoning, with improvements averaging up to 10% over the baseline.

Keywords: large language models; robotics; agentic workflows; reflection; semantic maps; object-centered planning

1. Introduction

Mobile robots are performing tasks of increasing complexity while also being deployed in an ever-expanding range of applications in fields such as home assistance, industry, healthcare, and education [1]. To effectively and autonomously complete these tasks, robots must have advanced cognitive capabilities enabling them to perceive, understand, and interact with their surroundings intelligently. Such capabilities, in turn, must be grounded in semantically rich representations of the environment, such as *semantic maps* [2–5]. These models encapsulate information about both the physical elements within the workspace (e.g., objects, rooms, etc.) and their associated semantics (e.g., properties, functionalities, relationships, etc.), providing robots with a comprehensive understanding of their environment and how to interact with it. For example, in a household environment, this information may include that kitchens are rooms for cooking (functionality) that contain appliances such as stoves and ovens (relationships), that refrigerators are used to store perishable food (functionality), and have a box-like shape (property), or that dining tables and chairs are used together for eating (relationships and functionality).

To accomplish specific tasks, robots leverage these representations to reason about the factual data acquired from the workspace and their semantics, inferring a plan—a process formally known as *planning*. If the planning focuses on the identification of the most relevant objects for the task, it can be referred to as *object-centered planning*. This requires reasoning about objects' functionality and properties to guide effective decisions. For instance, a robot in a kitchen tasked with preparing toast for breakfast must identify and localize the objects relevant to this task, such as a toaster, bread, etc.

Traditionally, ontologies [6] have been used to formally codify semantic knowledge using languages such as RDF [7] or OWL [8], which can be accessed through query languages such as SPARQL [9]. The retrieved information can then be used by rule-based languages like SWRL [10] (e.g. , `Bread(?b) and Toaster(?t) -> canToast(?t,?b)`) or planning languages like PDDL [11] to generate plans to solve the task at hand. Notice that these approaches rely on predefined knowledge and rules, meaning that anything beyond this scope lies beyond the robot's understanding; the robot is endowed with a *closed knowledge base*. This results in a lack of adaptability so that the robot can only handle predefined objects, tasks, and scenarios, which also leads to scalability issues as the range of tasks and the complexity of the environment increase.

Recent research has explored the utilization of *Large Language Models* [12–16] in robotic planning pipelines to address these limitations [17,18]. Trained on vast amounts of real-world data, these models offer a more flexible and comprehensive understanding of the diverse situations in which a robot may find itself. In this approach, the careful definition of LLM inputs –commonly called *prompts*– is essential, as prompt engineering techniques can significantly improve model performance [19]. In this context, prompts are not fixed, but templates where managed information (e.g. , a semantic map, a user query, etc.) is inserted. However, a major disadvantage of relying on LLMs is their tendency to produce incorrect, inaccurate, or invented information, a phenomenon known as hallucination [20] that can potentially lead to erroneous robot behavior.

To mitigate the risk of hallucinations, recent advancements have introduced *agentic workflows* to enhance the quality of responses produced by these LLM-driven systems. These workflows allow LLM systems to incorporate actions beyond simply generating text. For example, the *Tool Use* agentic workflow [21–23] provides models with access to external resources (e.g. databases, calculators, sensors, APIs) to supplement the model with more reliable sources of information than its mere internal knowledge. Another particularly promising agentic workflow is *Reflection* [24,25], where the model autonomously evaluates its outputs by reviewing its responses, generating feedback and making refinements without relying on external resources. However, the effectiveness of this technique in the context of object-centered planning has yet to be thoroughly evaluated.

To address this gap, in this work, we present a *quantitative evaluation* of the impact of using reflection-based agentic workflows in state-of-the-art LLMs for planning tasks. Specifically, we first formally define the application of three agentic workflows for this task: (1) *Self-Reflection*, where the LLM independently reviews and refines its responses; (2) *Multi-Agent Reflection*, involving multiple models generating responses, generating feedback and collaborating; and (3) *LLM Ensemble*, which uses several LLMs to generate a set of responses and then an evaluator LLM to choose the one with the highest quality. Given the lack of a standard dataset for evaluating these techniques in the object-centered planning context, we also contribute a novel dataset consisting of semantic maps, predefined queries, and annotated responses. To set up this dataset, we generated semantic maps from five scenes each from the popular ScanNet [26] and SceneNN datasets [27] using Voxeland [28], a probabilistic framework to build semantic maps that account for the inherent uncertainty of the object detection process. The queries simulate real-world user requests to a robot equipped with a semantic map representation and are categorized into three types: *Descriptive*, *Affordance*, and *Negation*, each presenting varying levels of complexity. The responses were first generated using an LLM and then refined using a Graphical User Interface (GUI) to ensure quality assurance. Finally, the dataset is used to evaluate the formalized techniques against a baseline LLM, assessing their impact on response quality.

The results revealed that agentic workflows improve response quality, particularly for queries requiring complex reasoning. Specifically, the *Self-Reflection* workflow achieved a maximum improvement of 10%, *Multi-Agent Reflection* 9.67%, and *LLM Ensemble* 4.67%. We also analyzed how the complexity of the semantic map affects overall performance, finding an inverse relationship between the number of objects in the map and performance, which is mitigated by agentic workflows compared to the baseline. Finally, we present the results of a root cause analysis for incorrect responses.

The dataset and the implementation used in our experiments are publicly available at <https://github.com/MAPIRlab/llm-robotics-reflection.git>.

2. Related Work

Semantic mapping refers to the process of linking factual information of elements in the robot workspace, typically collected through onboard sensors (RGB-D cameras, laser scanners, etc.) and aggregated in the form of some spatial primitive (occupancy grid maps, point clouds, bounding boxes, voxelized maps, etc.), with their semantics (properties, functionalities, relations, etc.) [29,30]. When this process focuses on modeling the objects within the workspace, it is called *object-oriented semantic mapping* [31].

Based on how semantic information is integrated into the representation of the workspace, semantic mapping approaches are divided into two groups: dense semantic mapping, which annotates each geometric primitive (e.g. a 3D point, a voxel, etc.) independently with its semantics [32,33], and instance-level semantic mapping, which represents objects as individual instances [34,35]. In either case, modern semantic mapping pipelines incorporate deep learning-based techniques to detect object categories [36] or instances [37] in input images. Recent works are replacing or complementing these networks with large vision-language models (LVLMs), enabling them to detect object categories beyond those considered in the network fitting [17].

In this work, we rely on Voxeland [28], a state-of-the-art probabilistic framework for building instance-aware semantic maps that can integrate any modern technique for object detection. This way, we evaluate the considered agentic workflows under realistic conditions, that is, with semantic maps built from comprehensive datasets using state-of-the-art techniques.

Semantic map exploitation for planning using LLMs involves translating natural language prompts into robot action plans, an approach that has become a growing trend due to the arrival of LLMs and their outstanding reasoning capabilities. Early studies, such as [11,38–40] exploited LLMs in semantic representations for this purpose though they are limited to single-room scenes. More recently, Sayplan [18] addresses this limitation by focusing on large-scale scenes, enabling LLM-based planning in more complex environments. Given a pre-built semantic map and a user query, SayPlan uses LLMs to identify the most relevant subset of the semantic map for the task, plan the robot's actions, and re-plan if the initial plan fails. One of the limitations of these approaches is the reliance on LLMs and their responses, which are known to sometimes include incorrect, inaccurate, or fabricated information, resulting in erroneous robot behavior.

In this work, we consider the simplified problem of object-centered planning, with responses limited to a list of suitable objects from the semantic map for the query, and quantitatively evaluate the impact of agentic workflows when dealing with this kind of tasks.

LLM response improvement has been an active research area since the advent of language models. Early works identified a misalignment between the LLM's primary objective, predicting the next token in a sentence, and the user's goal of having the LLM accurately follow instructions [41]. To address this gap, researchers initially relied on external supervision which required large training datasets [42] or costly human annotations [41], usually followed by a fine-tuning process of the model. Subsequently, more flexible approaches emerged, such as *Chain-of-Thought* [43] prompting, which basically encourages the model to generate intermediate reasoning steps before generating a response. Prompting techniques, which involve only modifying input prompts following a certain approach and no extensive training, have proven to be highly effective in improving response quality [44].

Further developments have evolved from prompting techniques to *agentic workflows*, which are structured, more complex processes that guide model responses through iterative refinements. One notable contribution in this area is Self-Refine [24], a method in which the model iteratively improves its outputs based on self-generated feedback. This work, along with others [21,25,45], has demonstrated the importance of LLMs reviewing their responses to generate refinements, a verbal reinforcement learning technique commonly referred to as *Reflection*. Complementing reflection-based methods,

multi-agent frameworks represent another trending line of research for enhancing LLM response quality [46–48]. In these workflows, a global task is distributed among several LLM-based agents, each assigned a specific role and responsible for completing certain sub-tasks. To accomplish their sub-task, they use their own reasoning capabilities and communicate with other agents, which converts the generation of the final response into a collaborative refinement, and may result in the overall system solving complex problems that it could not solve before. This multi-agent paradigm has proven to be especially effective in software development [46,47] due to its similarity with the collaborative nature of a real software project. Other works have generalized the multi-agent paradigm, making it adaptable to a wide range of applications [48].

In this work, we formally define and evaluate various forms of agentic workflows, especially those based on reflection and multi-agent communication.

3. Method

Simply put, an object-oriented semantic map contains a set of objects defined by physical features (geometric properties like position or size, appearance, etc.) and semantic information (e.g. object category, descriptive captions, relationships). Based on a scene's pre-built semantic map, an object-centered planning system receives a natural language query as *input* and responds with a list of the objects in the scene most related to the query as *output*. The addressed problem is formulated in more detail in Section 3.1.

Integrating an LLM into object-centered planning offers advantages such as enhanced reasoning capabilities, access to a broader open knowledge base with real-world information, and natural language understanding (see Section 3.2). However, this approach places significant trust in the model's responses, which may be incorrect, inaccurate, or fabricated. For this reason, it is advisable to refine the responses obtained, and one of the most promising methods to do so is to use LLM agentic workflows. The next sections formally define the three agentic workflow variants under study. Briefly, by incorporating the *Self-Reflection* workflow (see Section 3.3) the model generates feedback on its own answers and then refines them based on this feedback. Implementing the *Multiagent Reflection* workflow (see Section 3.4), three LLM-based agents work collaboratively: one generates the answers, another provides feedback, and the third agent refines the answers according to the feedback. In the *LLM Ensemble* workflow (see Section 3.5) multiple LLMs produce responses, and another is tasked with selecting the most appropriate from these options. All these workflows are summarized in Figure 1.

Finally, we describe the context for quantitatively evaluating these workflows performance (see Section 3.6), which includes the presentation of a novel dataset specifically tailored for this task (see Section 5).

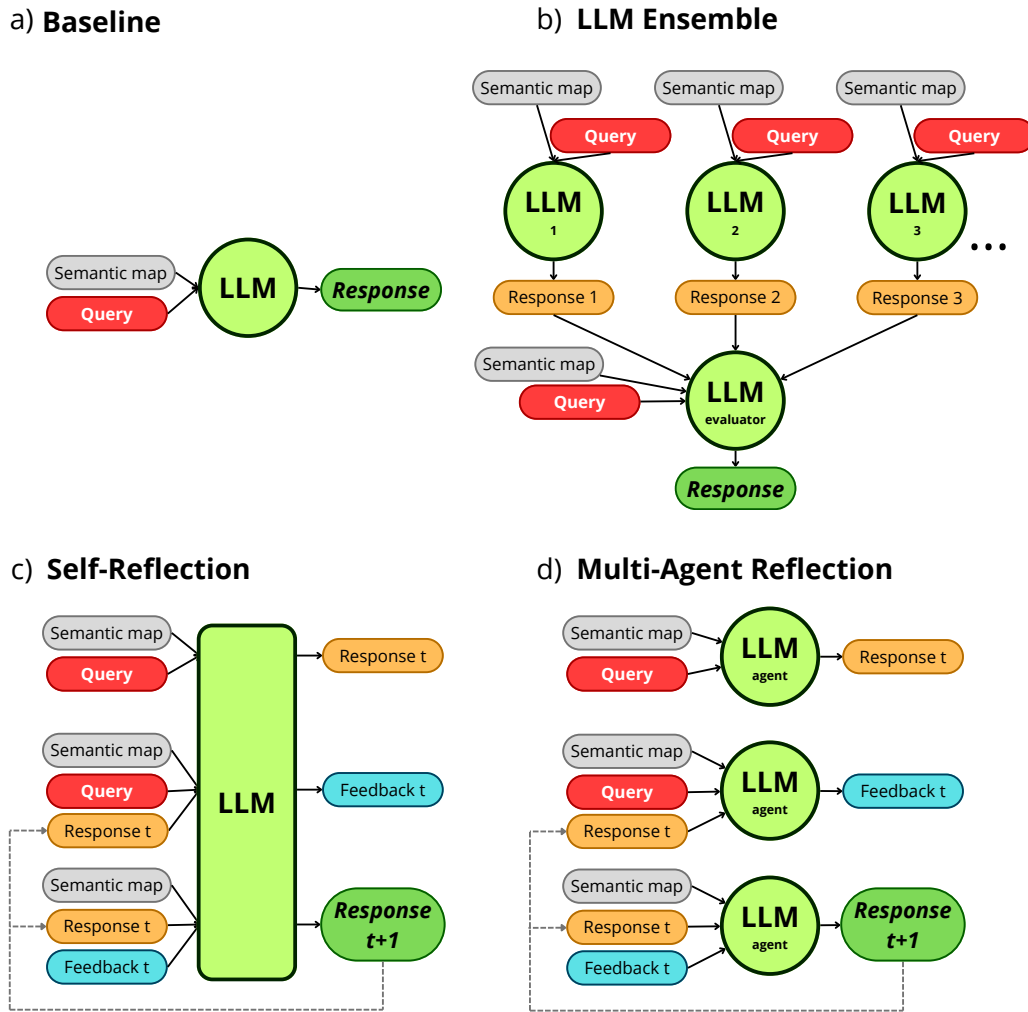


Figure 1. Overview of the baseline approach (a) and the presented workflows: *LLM Ensemble* (b), *Self-Reflection* (c), and *Multi-Agent Reflection*. Semantic maps are represented in gray, queries in red, intermediate responses in orange, feedback on responses in blue, and enhanced responses in green.

3.1. Problem Formulation

In essence, a semantic map of a 3D environment \mathbf{m} can be defined as:

$$\mathbf{m} = \{\mathbf{o}_1, \dots, \mathbf{o}_N \mid \mathbf{o}_i = (c_i, d_i, l_i, p_i), \forall i \in \{1, \dots, N\}\}$$

where:

- $c_i \in \mathbb{R}^3$ represents the i -th object's localization (e.g. the local center of coordinates of the bounding box).
- $d_i \in \mathbb{R}^3$ represents the i -th object's dimensions (e.g. the dimension of the bounding box).
- $l_i \in \mathcal{L}$ is the semantic label assigned to the i -th object, being \mathcal{L} the set of all possible semantic classes (e.g. table, chair, vase, etc.).
- p_i is a list of possible additional properties describing the i -th object.

Note that p_i can be used to store supplementary information relevant to the problem at hand, such as uncertainties in object classifications, multiple semantic labels each with a confidence score, or relationships among objects in the semantic map. Voxeland, the technique we have resorted to build semantic maps, is further described in Section 4.1.

In this context, given a semantic map of a working environment \mathbf{m} and a natural language query x , a system performing object-centered planning π generates a response y consisting of a list of objects from the semantic map related to that query:

$$y = \pi(\mathbf{m}, x) = [\mathbf{o}_j \mid j \in J \subseteq I] \quad (1)$$

where I is the set of indices of objects in the environment and $J \subseteq I$ denotes the selected objects' indices. This list of objects should be ordered according to each object's relevance to the query x , with the most convenient objects first. Notice that, as J can be \emptyset , the system can conclude that no object in the scene is convenient for query x .

3.2. Baseline

An object-centered planning system can leverage the advanced reasoning abilities of LLMs to enhance their understanding of the scene and provide more accurate answers. A baseline approach for implementing this would be considering an arbitrary LLM, \mathcal{M} , which plays the role of π and generates a response when provided with a prompt where all the information regarding the problem is included. More specifically, this prompt, p_{plan} , should embed both the semantic map \mathbf{m} and the natural language query x (see Figure 1(a)). It should also be specifically designed for the task at hand, instructing the LLM to generate a response consisting of a list with the objects most convenient for acting x . To enable LLMs to work with and understand a semantic map \mathbf{m} , we must represent it in a way that is compatible with these models. To this end, a commonly used approach is representing maps in JSON format [17], \mathbf{m}_{JSON} . The specific structure of these JSON files is not crucial, as long as a description of its particular structure is provided to the LLMs in the prompts. Thus, Eq. (1) is particularized for the baseline approach as follows:

$$y_0 = \mathcal{M}(p_{\text{plan}} \parallel \mathbf{m}_{\text{JSON}} \parallel x) \quad (2)$$

Note that \parallel denotes concatenation.

When working with LLMs, it is both possible and desirable to request more information from models than is strictly necessary to accomplish the task, i.e. the list of objects. As in [17], for each request, we ask the model to generate the following additional information:

- Inferred query: a short interpretation of the query.
- Query achievable: whether or not the user query is achievable using the objects in the semantic map.
- Explanation: brief explanation of what are the most relevant objects, and how they can be used to achieve the task.

Asking for this extra information can improve response quality [43] and provides insight into the model's reasoning process, which is essential for the iterative refinement of prompts.

Examples. To enhance the LLM's understanding of the task, few-shot prompting –also known as "in-context learning"– [49] has proven to be an effective technique across various domains. This technique provides the model with k in-context examples of the task, where each example pairs an input with the desired model's output. In our case, examples within p_{plan} follow the format $\langle \mathbf{m}_k, x_k \rightarrow y_k \rangle$, where \mathbf{m}_k represents a small semantic map tailored for the example, x_k is an example query, \rightarrow denotes model text generation, and y_k is the expected response.

3.3. Self-Reflection

Building upon the baseline approach, the first of the agentic workflows considered in this work for evaluation is the popular *Self-Reflection* [24,25]. This workflow extends the operation of the baseline by including a second step in which the LLM *reflects* on the initial response to provide feedback. The latter is then exploited to *refine* and improve the initial response (see Figure 1(c)). By reflecting on the plan offered in the first instance the LLM can refine it, by providing clearer explanations or even correct errors that would have resulted in the erroneous execution of the task, for example.

The reflection and refinement prompts are crucial in this workflow and must be designed accordingly. The reflection prompt p_{reflect} instructs the model to evaluate a previously generated response

regarding three evaluation criteria: correctness, relevance, and clarity. Correctness ensures that the inferred query and all relevant objects have been correctly identified and that the response is in the expected format. Relevance, on the other hand, assesses that objects are correctly ordered depending on relevance, none of them is omitted, and none is over. Finally, clarity assesses that the explanations provided are simple and unambiguous. Based on this evaluation, the model is asked to provide a list of clear, concise, and *actionable* suggestions for improving the preliminary response. Next, the refinement prompt p_{refine} requires the model to return a revised response, with the same format as the original, but incorporating the modifications suggested in the reflection step. This self-reflection process can be repeated iteratively. Notice that these prompts are intentionally interconnected to help the LLM better understand the flow of information in the task at hand. This means that p_{reflect} includes not only the initial response but also a brief summary of the LLM prompt that produced it. Similarly, p_{refine} includes short summaries of initial response and feedback generation prompts.

Thus, after the first response is generated through Eq. (2), the process of self-reflection comes into play using p_{reflect} , where the query provided by the user x , the semantic map \mathbf{m} , and the previous response y_t (y_0 in the first iteration) are inserted. With this input, the model \mathcal{M} generates feedback on the t -th response, fb_t :

$$fb_t = \mathcal{M}(p_{\text{reflect}} \parallel \mathbf{m}_{\text{JSON}} \parallel x \parallel y_t) \quad (3)$$

Once the feedback fb_t is generated, the refinement prompt p_{refine} is built using both the previous response y_t and its corresponding feedback fb_t . The output of this step is an enhanced response, y_{t+1} :

$$y_{t+1} = \mathcal{M}(p_{\text{refine}} \parallel \mathbf{m}_{\text{JSON}} \parallel y_t \parallel fb_t) \quad (4)$$

Regarding iterations, this process can be performed a fixed number of times or may depend on a different stopping condition. In case a stop condition is used, it could be external (e.g. the relevant objects have been the same in the last two response refinements) or requested in p_{reflect} [24]. The latter option would entail modifying p_{reflect} to ask the LLM to report whether the reflected answer is acceptable enough.

Memory. Given the iterative nature of this process, it is possible to make LLMs recall previous requests and responses by preserving the conversation history. In our context, this technique can be useful in both reflection and refinement stages. By reflecting and recalling previous messages, the LLM can avoid generating feedback on aspects that have already been evaluated. Similarly, refining and recalling previous interactions prevents the LLM from making mistakes that have already been corrected. With this approach, every time the LLM is going to generate feedback fb_i on response y_i , all previous responses and feedback generated on them from $(y_0 \rightarrow fb_0)$ to $(y_{t-1} \rightarrow fb_{t-1})$ will be recalled, so Eq. (3) changes to:

$$fb_t = \mathcal{M}(p_{\text{reflect}} \parallel \mathbf{m}_{\text{JSON}} \parallel x \parallel y_0 \rightarrow fb_0 \parallel \dots \parallel y_{t-1} \rightarrow fb_{t-1} \parallel y_t) \quad (5)$$

Likewise, on each response refinement y_{t+1} generated from response y_t and feedback fb_t , all previous preliminary responses, feedback and refinements from $(y_0, fb_0 \rightarrow y_1)$ to $(y_{i-1}, fb_{i-1} \rightarrow y_i)$ will be recalled, so Eq. (4) changes to:

$$y_{t+1} = \mathcal{M}(p_{\text{refine}} \parallel \mathbf{m}_{\text{JSON}} \parallel y_0, fb_0 \rightarrow y_1 \parallel \dots \parallel y_{i-1}, fb_{i-1} \rightarrow y_i \parallel y_t, fb_t) \quad (6)$$

This type of memory is often referred to as short-term memory and primarily aids context-aware decision-making [46]. However, its primary limitation is that information is lost when the conversation history exceeds the LLM's context window. When this occurs, the LLM "forgets" the earlier interactions. This problem could be solved with long-term memory techniques, e.g., every few iterations, an LLM could summarize the key points from previous iterations of the conversation and incorporate this information into the workflow prompts. Despite this, the fact that early reflection iterations contribute most to improving response quality [24] makes our process not to require a large number of iterations.

Furthermore, state-of-the-art LLM context windows are continuously expanding, so this short-term memory is enough in our case.

Examples. Same as for p_{plan} , prompts p_{reflect} and p_{refine} can also incorporate in-context examples to improve problem understanding by the model. In the case of p_{reflect} , examples will be of the form $\langle \mathbf{m}_k, x_k, y_k \rightarrow \text{fb}_k \rangle$, with y_k representing a response to generate feedback on, and fb_k being the expected feedback. In the case of p_{refine} , examples will be of the form $\langle \mathbf{m}_k, y_k, \text{fb}_k \rightarrow y_k^c \rangle$, with y_k^c representing the expected refined response. Again, \mathbf{m}_k represents a reduced semantic map tailored for the example at hand.

3.4. Multi-Agent Reflection

The *Multi-Agent Reflection* workflow utilizes three different LLMs acting as agents: one to generate initial planning responses (planner agent), another to provide constructive feedback on the responses (feedback agent), and a third to refine responses based on the previously generated feedback (refinement agent) (see Figure 1(d)).

Prompts with which each agent is equipped in this workflow differ slightly from those explained in Section 3.3. While the prompts in the *Self-Reflection* workflow instruct LLMs to perform specific tasks, the prompts in the *Multi-Agent Reflection* workflow instruct each model to adopt a particular role, functioning as an independent agent responsible for a specific part of the process. For example, while p_{plan} is framed as "Generate a plan for this query on this semantic map...", the equivalent prompt used in this workflow p_{plan}^a is framed as "You are an agent in charge of generating responses to queries based on this semantic map...". The same applies for p_{reflect} with p_{reflect}^a and p_{refine} with p_{refine}^a .

Thus, the planner agent will behave similarly to Eq. (2), with the particularity of prompt p_{plan}^a :

$$y_0 = \mathcal{M}(p_{\text{plan}}^a \| \mathbf{m}_{\text{JSON}} \| x) \quad (7)$$

In turn, the feedback agent will act as Eq. (3) but using p_{reflect}^a :

$$\text{fb}_t = \mathcal{M}(p_{\text{reflect}}^a \| \mathbf{m}_{\text{JSON}} \| x \| y_t) \quad (8)$$

And the refinement agent behaves as Eq. (4), but using p_{refine}^a :

$$y_{t+1} = \mathcal{M}(p_{\text{refine}}^a \| \mathbf{m}_{\text{JSON}} \| y_t, \text{fb}_t) \quad (9)$$

As derived from Eq. (8) and Eq. (9), feedback and refinement agents can iterate until the specified stopping condition is met, as described in Section 3.3.

Memory. Although the above formulation does not include short-term memory, this is usually a characteristic of LLM-based agents [46,48]. It can be easily implemented by incorporating the prompts p_{reflect}^a and p_{refine}^a in Eq. (5) and Eq. (6).

Examples. Since the task the agents should accomplish is the same as in Self-Reflection, in-context examples in prompts will have the same structure, and can be reused without modification.

3.5. LLM Ensemble

In the *LLM Ensemble* workflow, multiple LLMs generate different responses to a query based on a semantic map. The best response from these options is then selected based on specific criteria (see Figure 1(b)). Each LLM on the ensemble may have different characteristics (e.g. architecture, model size, training data), though this is not essential, as the same LLM can produce various outputs for the same inputs due to the inherent randomness in its generation process [12]. While the planning generation prompt p_{plan} can remain unchanged, this workflow requires an additional prompt p_{choose} which instructs the model to select the most appropriate response for the task from a list. This selection prompt should only include the responses without indicating which model generated each answer, to avoid potential bias [50].

So, supposing we have a number n of arbitrary LLMs \mathcal{M}_k , the first step is to generating a planning response for each one, which slightly modifies Eq. (2):

$$y_k = \mathcal{M}_k(\mathcal{P}_{\text{plan}} \parallel \mathbf{m}_{\text{JSON}} \parallel x) \quad (10)$$

Then an evaluator LLM \mathcal{M}_c , that is not necessarily different from the ones that generated the answers, is passed $\mathcal{P}_{\text{choose}}$ with the previous responses inserted, to generate the definitive answer y_* .

$$y_* = \mathcal{M}_c(\mathcal{P}_{\text{choose}} \parallel \mathbf{m}_{\text{JSON}} \parallel \{y_k\}_{k=1}^n) \quad (11)$$

Note that here the response is not refined, but exactly one of the generated by the models in the ensemble: $y_* \in \{y_k\}_{k=1}^n$.

Memory. As this process is not iterative, the LLM does not need memory to recall previous utterances.

Examples. Same as for previous prompts, prompt $\mathcal{P}_{\text{choose}}$ could also implement examples, that would be of the form $\langle \mathbf{m}_k, y_{k,1}, y_{k,2}, \dots, y_{k,n} \rightarrow y_k^c \rangle$, with \mathbf{m}_k representing a semantic map tailored for the example, $y_{k,1}, y_{k,2}, \dots, y_{k,n}$ being the set of n responses returned by the the ensemble from which one should be selected, and y_k^c being the chosen answer.

3.6. Agentic Workflows Evaluation Context

To quantitatively evaluate the improvement in responses using these workflows (*Self-Reflection*, *Multi-Agent Reflection*, and *LLM Ensemble*) with respect to the base model, we rely on a custom-designed dataset tailored for object-centered planning tasks. This dataset is specifically designed to assess LLMs' reasoning abilities in terms of spatial understanding, object functionality, and negated instructions. The following section details the construction of this dataset, including the design of the queries, the generation of the semantic maps, and the selection of the environments.

4. Dataset Description

The collected dataset for evaluating the agentic workflows comprises 30 queries that simulate requests that are likely to arise in an indoor working environment. All the created queries forming the set \mathcal{Q} are listed in Appendix A. Each query is intended to be answered by identifying the list of objects most suitable for completing the specified task. Thus, the ground truth contains the evaluation for all queries across all semantic maps \mathcal{SM} , that is, a list of objects to satisfy each query's requirements effectively. Formally, the dataset is defined as:

$$\mathcal{D} = \{(x^{(i)}, \mathbf{m}_{\text{JSON}}^{(j)}, r^{(i,j)}) \mid x^{(i)} \in \mathcal{Q}, \mathbf{m}_{\text{JSON}}^{(j)} \in \mathcal{SM}, 1 \leq i \leq m, 1 \leq j \leq n\} \quad (12)$$

where:

- $x^{(i)}$ is a query from the set of queries \mathcal{Q} with size m .
- $\mathbf{m}_{\text{JSON}}^{(j)}$ is a map in JSON format from the set of semantic maps \mathcal{SM} with size n .
- $r^{(i,j)} \subseteq \mathcal{O}^{(j)}$ is the response to query $x^{(i)}$ when evaluated on map $\mathbf{m}_{\text{JSON}}^{(j)}$, being $\mathcal{O}^{(j)}$ the set of all objects in said map.

4.1. Generating the Semantic Maps

To provide realistic data that could be produced/consumed during typical robot operations, we employ a state-of-the-art framework to generate semantic maps, rather than using artificial ground-truth maps that include the entire range of objects in a working environment even if the robot would be unable to perceive them.

Concretely, we rely on Voxeland [28], a probabilistic framework for incrementally building instance-aware semantic maps from a sequence of RGB-D images. Voxeland operates by generating a 3D, voxel-based reconstruction of the environment, in which each voxel holds a probability distribution

over the set of possible object instances within the map. Each object instance, in turn, is annotated with a probability distribution across the set of possible object categories. This probabilistic framework enables a more realistic representation, where the true category of each object instance needs to be interpreted in probabilistic terms, rather than a deterministic label, reflecting the uncertainty inherent to the object categorization [51,52].

Upon completion of the map generation, Voxeland outputs this information in two parts. First, it provides the voxel-based environment reconstruction, where each voxel is annotated with instance-level information, as shown in Figure 2. Second, Voxeland yields a JSON-structured text containing both geometric and semantic information (i.e. bounding boxes' centers and dimensions, object categorization in form of a probability distribution, number of observations, etc.) about each map instance. An example of this JSON output is shown below:

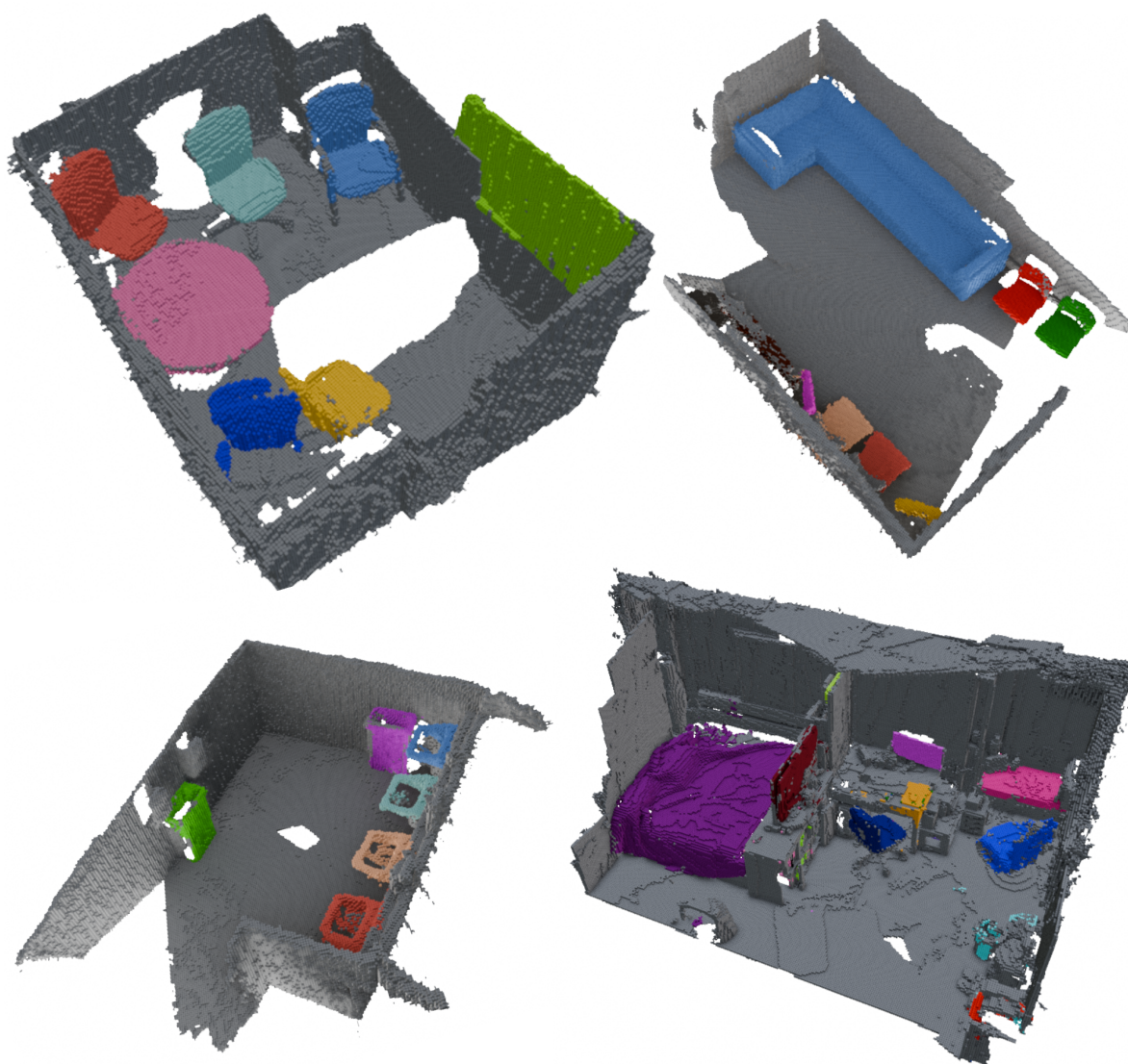


Figure 2. Voxel-based reconstructions generated by Voxeland for scenes from the ScanNet [26] dataset. Gray voxels represent uncategorized regions, while colored voxels indicate distinct object instances within the map.

```

{
  "instances": {
    "obj1": {
      "bbox": {
        "center": [3.30, 2.00, 0.36],
        "size": [0.23, 0.1, 0.05]
      },
      "n_observations": 147,
      "results": {"cell phone": 0.72, "remote": 0.26, "tie": 0.02}
    },
    ...
  }
}

```

This JSON-structured text represents the semantic map of the environment, denoted in this work as m_{JSON} , and serves as the input to our proposed method. Notice that when building the dataset we considered the most probable semantic label of each object as the right one (e.g. `cell phone` in the previous example).

4.2. Datasets Description

The scenes used to generate the semantic maps in our dataset, along with corresponding queries and ground-truth responses, come from two widely used publicly available datasets: ScanNet [26] and SceneNN [27]. Using two different datasets provides heterogeneity in the generated semantic maps, in terms of scene size and number of objects within it, which enriches the workflow evaluation.

ScanNet is a large-scale dataset consisting of annotated 3D reconstructions of indoor scenes. For each of its nearly 1,500 scenes, ScanNet provides RGB-D images, semantic annotations, and camera poses. From this dataset, we selected large scenes containing a high number of objects, specifically from five different one-story apartments: `scene_0000`, `scene_0101`, `scene_0392`, `scene_0515`, and `scene_0673`.

The SceneNN dataset also provides a collection of annotated 3D reconstructions of indoor scenes. It also offers RGB-D images, camera poses and ground-truth semantic reconstructions. Scenes selected from this dataset are generally smaller, with fewer objects than those in ScanNet, and come from a lounge (scene 011), offices (scenes 030, 078, 086), and a bedroom (scene 096).

4.3. Queries Design

Following the categorization in [17], queries in \mathcal{Q} are divided into three types according to the required reasoning: *descriptive*, *affordance*, and *negation*.

- Descriptive queries prompt the system to identify objects based on their category, some characteristics within a scene, or their spatial location. An example of this query type is `query_09` which states *"Is there a microwave that is close to an oven?"*. The evaluation of this type of queries provides an approximation of how the system will behave in simple tasks.
- Affordance queries evaluate the system's ability to infer the functionality and uses of objects in specific contexts. In general, they ask for objects to perform an action or to fill a need. An example of this is `query_15`, *"I am tired, where can I sit?"*. These queries assess application of real-world information in our context by LLMs.
- Negation queries test the model's ability to find a solution excluding specific objects or objects with specific features. An example is `query_23`, *"Tell me something I can sit on that is not a couch"*. These queries are especially interesting for the evaluation of workflows, as LLMs tend to perform poorly when inputted with prompts containing *negated instructions* [53].

In addition, each query type is further divided into two difficulty levels: *easy* and *difficult*.

4.4. Annotating Responses

The ground truth was generated by the authors of this work by evaluating each query in \mathcal{Q} against all the generated semantic maps in \mathcal{SM} . To accelerate this process, a GUI was developed. The GUI displayed semantic map objects and queries side by side and allowed for the dynamic input

of responses (see Figure A1). All the ground truth data can be found in structured files within the repository.

It is important to note that the semantic maps' ground truth was defined based on Voxeland's output, instead of using the dataset's original ground truth. In this way, each object is considered to be of the category for which Voxeland provided the most certainty.

An initial version of the response ground truth was generated using ChatGPT o1-preview. Later, all the responses were manually reviewed. During this review, the original scene reconstruction was examined using online tools that visualize 3D representations of the ScanNet¹ and the SceneNN² datasets.

5. Experimental Setup

To quantitatively analyze the impact on response quality of the presented workflows in an LLM-based system performing object-centered planning, we conducted experiments using a self-created dataset. This dataset comprises: i) a set of semantic maps, ii) a collection of queries related to these semantic maps that a user could formulate, and iii) the corresponding ground-truth responses for each semantic map-query pair. The semantic maps available in the dataset were generated using the state-of-the-art approach Voxeland [28] instead of using the ground-truth semantic maps (see Section 4.1). Voxeland addresses the semantic mapping of a scene by dealing with uncertainties in object classification, enabling us to implement a more realistic evaluation. The scenes from which Voxeland generated the semantic maps came from the publicly available datasets ScanNet and SceneNN (see Section 4.2). The ground-truth responses for each map-query pair consist of a ranked list of objects from the semantic map that are most relevant to the query. The list is ordered by relevance, with the most important objects appearing first and the least important ones appearing last. The proposed workflows were evaluated on all the scenes and queries in our dataset (see Section 4.2) using state-of-the-art LLMs from the Google Gemini family (see Section 5.1). For a comprehensive evaluation, we compare the responses generated by the workflows with the ground-truth data. Similarly to [17], for this evaluation, we employed four different metrics (see Section 5.2 for more details): *Top-1*, *Top-2*, *Top-3*, and *Top-Any*.

5.1. Implementation Details

The presented workflows were implemented using LLMs from the Google Gemini family [54], specifically *Gemini 1.0 Pro* (`gemini-1.0-pro`) and *Gemini 1.5 Pro* (`gemini-1.5-pro`). Among these, results generated by *Gemini 1.5 Pro* have proved more relevant for the evaluation due to its enhanced reasoning capabilities and improved contextual understanding. As these models are not publicly available, we accessed them via API calls. By using them in this way, we take advantage of fast text generation and avoid installing additional software. In addition, current proprietary LLMs are so far better than open-source alternatives³. On the other hand, all requests are charged a cost depending on the number of input and output tokens, which depends on the number of input and output tokens.

In the baseline approach, only *Gemini 1.5 Pro* was used to generate responses. In the *Self-Reflection* and *Multi-Agent Reflection* workflows *Gemini 1.5 Pro* was employed too, and the reflection process was performed in two iterations. The *LLM Ensemble* workflow was implemented with an ensemble of six *Gemini 1.5 Pro* models to generate answers. The selection of the best response was handled by a separate evaluator LLM, which also utilized *Gemini 1.5 Pro*.

Regarding examples in prompts, the planning prompts $p_{\text{plan}}, p_{\text{plan}}^a$ incorporated three examples, while the reflection prompts $p_{\text{reflect}}, p_{\text{reflect}}^a$ incorporated seven examples, the refinement prompt $p_{\text{refinement}}, p_{\text{refinement}}^a$ incorporated three examples, and the evaluator prompt p_{choose} incorporated no examples. All these prompts are designed to carefully follow the prompts engineering

¹ https://kaldir.vc.in.tum.de/scannet_browse/

² <https://hkust-vgd.ust.hk/scenenn/home/webgl/>

³ <https://artificialanalysis.ai/leaderboards/models>

recommendations provided by Google for the Gemini family models⁴. The *Self-Reflection* and the *Multiagent-Reflection* workflows were both implemented with memory on previous reflection iterations.

5.2. Metrics

The metrics used in this work aim to assess the degree of similarity between two ordered lists of objects, representing responses generated by a system performing object-centered planning. Therefore, each system's output for a given query within a specific semantic map, $y_\pi = \{\mathbf{o}_1, \dots, \mathbf{o}_{N_\pi}\}$, is compared to the annotated ground truth for the same query and semantic map, $y_{\text{GND}} = \{\mathbf{o}_1, \dots, \mathbf{o}_{N_{\text{GND}}}\}$, where N_π and N_{GND} denote the number of objects in the response of the system and in the ground truth, respectively. To evaluate the similarity between these two lists, the following metrics are employed:

- *Top-1* checks if the first object in the system's response, $y_\pi[1]$, appears in the ground truth, $y_{\text{GND}}[1]$, or if both lists are empty. It is defined as:

$$\text{Top-1} = \begin{cases} 1, & \text{if } y_\pi[1] \in y_{\text{GND}} \text{ or } (y_\pi = \emptyset \text{ and } y_{\text{GND}} = \emptyset) \\ 0, & \text{otherwise.} \end{cases}$$

- *Top-2* checks if any of the first two objects from the system response, $y_\pi[1]$ and $y_\pi[2]$, belong to the ground truth, $y_{\text{GND}}[1]$. It is defined as:

$$\text{Top-2} = \begin{cases} 1, & \text{if } y_\pi[i] \in y_{\text{GND}} \text{ for some } i \leq 2 \\ 0, & \text{otherwise.} \end{cases}$$

- *Top-3*, similar to *Top-2*, checks whether some of the first three objects in the system response belong to the ground truth:

$$\text{Top-3} = \begin{cases} 1, & \text{if } y_\pi[i] \in y_{\text{GND}} \text{ for some } i \leq 3 \\ 0, & \text{otherwise.} \end{cases}$$

- *Top-Any* measures whether any of the objects in the system's response appear in the ground truth. It is defined as:

$$\text{Top-Any} = \begin{cases} 1, & \text{if } y_\pi[i] \in y_{\text{GND}} \text{ for some } i \\ 0, & \text{otherwise.} \end{cases}$$

Note that these measures are cumulative. For example, if a query has a result for *Top-1*, it will also have results for *Top-2*, *Top-3*, and *Top-Any*, as objects included in a narrower ranking are inherently part of the broader rankings.

6. Evaluation

After obtaining the responses for each workflow over the dataset and evaluating them against the annotated ground truth, we present a detailed analysis of the results. First, compare performance across all workflows (see Section 6.1), showing that different workflows excel for specific query types while underperforming for others. Second, we evaluate the influence of semantic map complexity, which is determined by the number of objects within it, on overall workflow performance (see Section 6.2). The chosen scenes are heterogeneous, containing different types of objects and coming from different places, so the evaluation provides us with an insight into which kind of scenes the workflows work best with. Finally, we perform a qualitative analysis of errors produced by the reflection-based workflows (see Section 6.3) by manually selecting and analyzing some failure cases in which the reflection responses were outperformed by the baseline response. This evaluation helped identify LLM misunderstandings, enabling the refinement of prompts. The results show that reflection is the most effective approach,

⁴ <https://ai.google.dev/gemini-api/docs/prompting-intro>

especially when applied to queries that require more reasoning. An increase in the size of semantic maps has been found to impact workflow performance negatively, and reflection-based workflows sometimes tend to *over-reflect* in certain responses.

6.1. Workflows Comparison

We evaluated the performance of the proposed workflows on all semantic maps and queries. The overall performance is summarized in Table 1, while Table 2 presents results grouped by dataset of origin and query types.

For the baseline approach, results highlight notable differences in performance depending on query type; it performs best on *Descriptive* queries, followed by *Affordance* queries, while *Negation* queries show the lowest performance. As stated in Section 4.3, queries were divided into three categories, based on the type of reasoning needed to respond. *Descriptive* queries, which involve simply identifying objects by their names or by some properties, are relatively straightforward and require minimal reasoning, leading to higher scores across workflows. In contrast, *Affordance* queries require identifying objects suitable for a specified action, requiring more reasoning about object functionality and task relevance. This adds complexity and results in moderated performance. *Negation* queries involve finding objects that lack certain attributes or must be excluded from a given list. They demand advanced reasoning and the ability to exclude objects that might appear at first glance to be task-related, which is an area where LLMs tend to make more mistakes [53]. Consequently, these queries yield the lowest performance. Given the limitations of the baseline approach, some methods of improving response quality, such as the agentic workflows presented in this work, are essential for real-world robotics applications.

Among the proposed workflows, the *LLM Ensemble* method yields the most moderate improvement in response quality. While it provides some gains in *Affordance* queries (e.g. up to +20% in *Top-Any* on ScanNet), it has a negligible or even negative impact on *Descriptive* and *Negation* queries, which results in an overall gain of +4.67% in *Top-Any* performance, and no improvement in terms of *Top-1* performance. This suggests that generating multiple similar responses without iterative refinement may not sufficiently improve response quality. Additionally, as the length of the prompts increases—particularly due to the inclusion of semantic maps and examples—the probability distribution of potential responses becomes more constrained. In essence, providing more information like detailed instructions and examples narrows the range of plausible outputs, thereby reducing the variability and diversity of the model’s responses for the same input. Without diversity in responses, the evaluator LLM is forced to select the best of the answers from a list of similar copies, which limits the potential of this workflow.

Based on the results, workflows yielding the best performance are *Multi-Agent Reflection* (+9.67% of improvement in *Top-Any*) and *Self-Reflection* (+10.0% improvement in *Top-Any*), so the process of iteratively improving responses by reflecting on them proves to be effective when increasing the system’s response quality. Between these, the one offering the best performance varies depending on the dataset. In the *ScanNet* dataset, *Self-Reflection* shows slightly better improvements (up to +12% in *Top-Any* accuracy), while in the *SceneNN* dataset, *Multi-Agent Reflection* performs better (up to +9.33% in *Top-Any* accuracy).

Table 1. Overall performance of the presented workflows. In bold, the technique with the highest performance for each metric.

Method	Top-1	Top-2	Top-3	Top-Any
Base	66.60	68.67	69.33	69.67
LLM Ensemble	0.00	+2.67	+4.00	+4.67
Multi-Agent Reflection	+8.33	+8.33	+9.00	+9.67
Self-Reflection	+7.67	+8.33	+8.67	+10.00

In addition, reflection-based workflows demonstrate clear improvements in handling query types that require higher reasoning, such as *Affordance* and *Negative* queries. For instance, in *Negation* queries on the SceneNN dataset, *Multi-Agent Reflection* improves *Top-Any* accuracy by +18%. However, it may slightly reduce the performance on simpler query types like *Descriptive* queries. In the case of *Affordance* and *Negative* queries, initial LLM responses tend to misinterpret the query intent, leading to incorrect object suggestions. Then, the reflection process effectively identifies these misunderstandings, generates feedback, and enables refinements in subsequent iterations, benefiting the overall result. On the other hand, a satisfactory initial response is usually produced for *Descriptive* queries. During the reflection stage, these responses are sometimes unnecessarily refined or altered, which leads to *over-reflection*, which degrades quality. This issue is particularly evident when the ground truth specifies no relevant objects (i.e. an empty list). In such cases, the initial response correctly indicates the absence of related objects, but reflection may force the system to infer or fabricate objects that are not fully relevant to the query, resulting in erroneous behavior. These issues were identified through failure-case analysis (see Section 6.3), and to mitigate them, the prompts were refined to include instructions aimed at preventing such behaviors.

Table 2. Presented workflows performance grouped by dataset and query type. In bold, the technique with the highest performance for each metric.

Dataset	Agentic Workflow	Query Type	Top-1	Top-2	Top-3	Top-Any
ScanNet [26]	Base	Descriptive	78.00	82.00	84.00	84.00
		Affordance	52.00	58.00	60.00	60.00
		Negation	52.00	54.00	54.00	54.00
		Average	60.67	64.67	66.00	66.00
	LLM Ensemble	Descriptive	0.00	+2.00	0.00	+2.00
		Affordance	+8.00	+10.00	+18.00	+20.00
		Negation	-4.00	+6.00	+8.00	+8.00
		Average	+1.33	+6.00	+8.67	+10.00
	Multi-Agent Reflection	Descriptive	+4.00	+2.00	+2.00	+2.00
		Affordance	+8.00	+8.00	+10.00	+10.00
		Negation	+16.00	+16.00	+18.00	+18.00
		Average	+9.33	+8.67	+10.00	+10.00
	Self-Reflection	Descriptive	+8.00	+6.00	+4.00	+4.00
		Affordance	+10.00	+8.00	+12.00	+14.00
		Negation	+14.00	+18.00	+18.00	+18.00
		Average	+10.67	+10.67	+11.33	+12.00
SceneNN [27]	Base	Descriptive	86.00	86.00	86.00	86.00
		Affordance	72.00	76.00	76.00	78.00
		Negation	56.00	56.00	56.00	56.00
		Average	71.33	72.67	72.67	73.33
	LLM Ensemble	Descriptive	-4.00	-4.00	-4.00	-2.00
		Affordance	-6.00	-8.00	-8.00	-10.00
		Negation	+6.00	+10.00	+10.00	+10.00
		Average	-1.33	-0.67	-0.67	-0.67
	Multi-Agent Reflection	Descriptive	+6.00	+6.00	+6.00	+6.00
		Affordance	+2.00	+2.00	+2.00	+4.00
		Negation	+14.00	+16.00	+16.00	+18.00
		Average	+7.33	+8.00	+8.00	+9.33
	Self-Reflection	Descriptive	0.00	0.00	0.00	0.00
		Affordance	+2.00	+2.00	+2.00	+8.00
		Negation	+12.00	+16.00	+16.00	+16.00
		Average	+4.67	+6.00	+6.00	+8.00

Additionally, a noticeable difference in overall performance exists between the datasets: results on ScanNet are lower than those on SceneNN. As stated in Section 4.2, these datasets are heterogeneous. Specifically, ScanNet comprises larger scenes with more objects and richer semantic diversity, while SceneNN provides smaller scenes with fewer objects and a more limited variety. To check if this first intuition was right, an analysis of performance depending on semantic map sizes has been conducted in Section 6.2.

Even though the presented workflows show improvements in response quality, the overall performance of the system is still moderate. One reason for this could be the inclusion of highly challenging queries, such as those requiring information beyond what is available in the semantic map or inferable by an LLM. For instance, query_16 asks "Where can I sit and observe the entire view of the room without obstruction?". Answering this query correctly requires not only understanding of the objects in the scene, and their function, shape, etc., but also considering spatial details such as the layout of walls, doors, and other structural elements. While this information is inherently considered in the ground truth, it is not provided to the model. Although these queries are deliberately challenging for current LLM capabilities and information available in our setting, their inclusion ensures that the dataset can still be used to evaluate future approaches and models.

On average, the time taken by the described systems to respond to all queries on a semantic map was: 3 m 3 s for the baseline approach (30 requests, 6.12 s/request), 21 m 26 s for the *Self-Reflection* workflow (150 requests, 8.58 s/request), 21 m 44 s for the *Multi-Agent Reflection* workflow (150 requests, 8.70 s/request), and 20 m 37 s for the *LLM Ensemble* workflow (210 requests, 5.89 s/request).

6.2. Impact of Semantic Map Complexity

This section aims to analyze how the performance of the considered object-centered systems is influenced by the complexity of the semantic map on which they operate. To this end, Figure 3 shows the average performance of the proposed workflows, in terms of the *Top-Any* metric, across all semantic maps' complexity. Note that complexity is quantified by the number of objects present in the scene, although, as discussed below, other factors may also contribute to it.

An inverse relationship is evident between the number of objects in a semantic map and the *Top-Any* metric performance. This trend is consistent across other metrics, though only the *Top-Any* metric is presented for simplicity. The decline in performance is observed in all of the presented workflows but is greater in the baseline approach. In contrast, reflection-based workflows, such as *Self-Reflection* and *Multi-Agent Reflection*, show a less pronounced decline as the number of objects in the semantic maps increases, with minimal differences between these two workflows. Despite the overall trend, some exceptions are observed. The most significant one may be the semantic map for scenenn_030 (SceneNN dataset), which contains a relatively high number of objects (26) yet achieves relatively strong performance on the *Top-Any* metric. Further analysis revealed that this map, though containing a large number of objects, included only five different object categories, making it highly redundant and exhibiting a poor semantic variety. Because of this redundancy, the ground truth includes a wider range of response objects in queries, especially in *Descriptive* and *Affordance* ones, where results are benefited. For example, if a query asks to find a book in the scene, and the scene contains 10 books, the LLM has an easier time finding an answer, since by pointing to 1 of those ten the answer will be marked as correct.

One potential cause of the observed negative trend in results as semantic map size increases is the relation between the number of objects in the map and the prompt length. As explained in Section 3, semantic maps consist of JSON files containing object information, which are directly included in prompts previously explaining their structure. Thus, when the number of objects increases, the size of the JSON file –and consequently the prompt length– also grows. Working with LLMs and large contexts presents some challenges. Specifically the "lost in the middle effect" [55] problem suggests that LLMs tend to perform better with information located at the beginning or end of prompts, while information in the middle is often forgotten or overlooked. Although in this work we have placed

semantic maps at the end of prompts following prompt engineering best practices, the growing length of prompts due to semantic maps may still worsen this effect, leading to poorer performance.

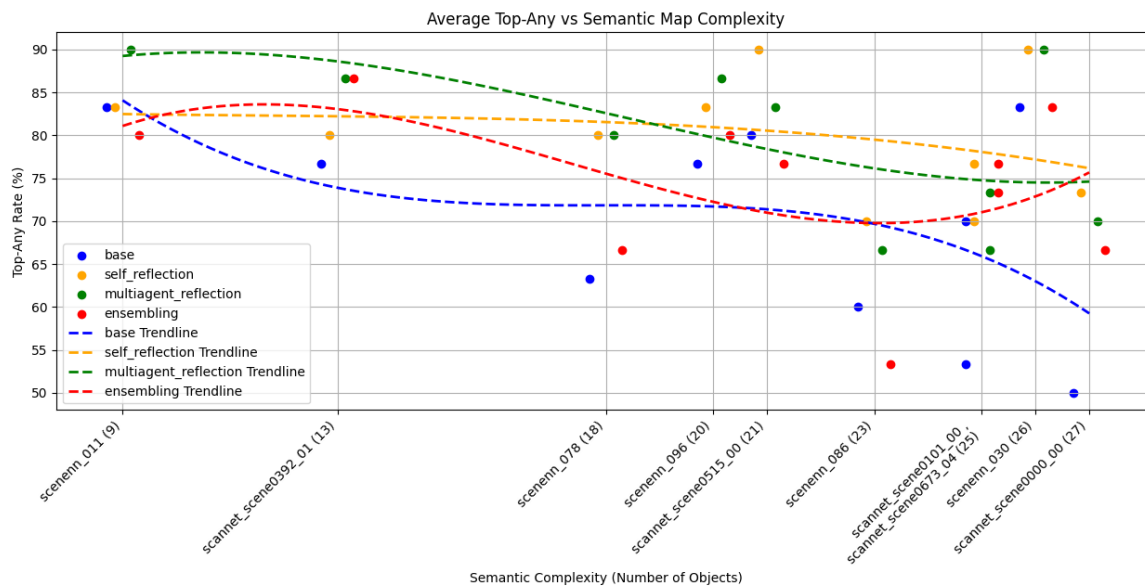


Figure 3. Average *Top-Any* rate (%) as a function of semantic map complexity, measured by the number of objects in the map. Each workflow is color-coded: blue (baseline approach), yellow (*Self-Reflection*), green (*Multi-Agent Reflection*), and red (*LLM Ensemble*). The dashed lines indicate the fitted third-degree polynomial regression.

6.3. Errors Qualitative Analysis

To better understand the performance of agentic workflows implementing reflection, a detailed error analysis was performed on the *Self-Reflection* and the *Multi-Agent Reflection* workflows. Specifically, we examined cases in which the baseline approach outperformed the reflection-based methods were selected and the cause of the error was manually identified. A total of 17 such cases were observed.

The most frequent error (70%) involved ground-truth queries with an empty list of objects. In these cases, the first planning stage correctly outputted an empty list, but the reflection process incorrectly added objects by *over-reflecting* on the result.

An example of this behavior is the result provided by the *Self-Reflection* workflow for query_20 ("What can I sit on that is near a book?") in the semantic map *scene0392* from ScanNet. Here, the baseline approach and the first iteration of the *Self-Reflection* workflow provided very similar responses, indicating that no object in the scene was available to accomplish the task. The reason for the response was that no books were present in the semantic map, which is correct:

```
{
  "inferred_query": "Identify a seat near a book.",
  "query_achievable": false,
  "relevant_objects": [],
  "explanation": "The provided semantic map does not contain any objects
classified as 'book'. Therefore, it is not possible to determine
which seat is near a book."
}
```

However, the reflection iterations pointed out that, while it is true that there is no directly related object in the scene, the task could be achieved using indirectly related objects:

1. Comments on Correctness. Score: 2/10

- Query achievable: Minor error. While there's no "book" object, the query

could still be achieved if other objects could be reasonably interpreted as suitable for sitting with a book. For example, a dining table might imply the presence of chairs nearby. However, the response completely dismisses the possibility.

[...]

This reasoning is erroneous, as it erroneously suggests that indirect relationships could justify including objects like chairs near a dining table, even though the absence of a book rendered the query unfeasible. This flawed reasoning led to a response that included unrelated objects:

```
{
  "inferred_query": "Identify a seat near a book.",
  "query_achievable": "partially",
  "relevant_objects": [
    "obj2",
    "obj20",
    "obj42"
  ],
  "explanation": "While there are no books explicitly identified, 'obj2' (dining table) suggests a possible reading location. Objects 'obj20' and 'obj42', identified as 'chair', are near the table and could be suitable for sitting. 'obj20' is located approximately 0.29 meters from the table's center. 'obj42' is positioned near the center of the table at a distance of 0.17 meters."
}
```

Note that, since the ground-truth result for this query is an empty list, the baseline response counts as a *Top-1* hit, while the refined reflection-based response does not count toward any metrics.

Other errors involved the removal of objects marked as relevant from the initial response after reflecting on it (12%) and an incorrect reordering of relevant objects (6%). These kinds of errors could be caused by the LLM misunderstanding the instruction of prompts and could be mitigated by providing clearer explanations or even including similar examples.

More challenging issues, such as object hallucinations (6%) or hallucinations when considering a clearly incorrect response as correct (6%) were also observed. Addressing these errors may require implementing more sophisticated workflows or even using better-performing LLMs.

7. Conclusions

This work arose from the need to quantitatively evaluate the impact of using agentic workflows in Large Language Models (LLMs) when performing object-centered planning in robotics. In this context, an LLM is equipped with a semantic map of the working environment and has to respond to natural language queries about it. Its responses consist of a list of objects, that are ordered depending on their relevance to the query.

We formalized and implemented three agentic workflows besides the baseline model: (1) *Self-Reflection*, where the LLM independently reviews and refines its responses, (2) *Multi-Agent Reflection*, involving multiple LLM-based agents generating responses, providing feedback, and collaborating to improve the output, and (3) *LLM Ensemble*, which uses several LLMs to generate responses, then employs an evaluator LLM to select the most appropriate one.

To perform the quantitative evaluation, we created a publicly available dataset comprising semantic maps, natural language queries, and manually annotated responses for each query and semantic map. Semantic maps were generated from scenes in the popular ScanNet and SceneNN datasets using the Voxeland semantic mapping framework. Queries were designed to simulate real-world user requests and were divided by the type of reasoning required to respond, as well as by difficulty.

Experiments show that reflection-based agentic workflows, *Self-Reflection* and *Multi-Agent Reflection*, generally improve LLM response quality in this context. On average, *Self-Reflection* outperforms the baseline by up to 12% on ScanNet and 8% on SceneNN, while *Multi-Agent Reflection* achieves improvements of 10% and 9.33%, respectively. The *LLM Ensemble* workflow yields more moderate improvements than the reflection-based approaches, reaching up to a 10% enhancement on ScanNet but showing no significant improvement on SceneNN. The response enhancement caused by reflection is more evident in queries requiring higher levels of reasoning while being less noticeable in the simplest and easiest queries. Thus, the reflection process effectively identifies misunderstandings in the initial responses, enabling refinements in subsequent iterations. Furthermore, the analysis revealed that workflows' performance is influenced by the complexity of the semantic maps, in terms of the number of objects present in a scene and their semantic diversity. Larger maps with more objects tend to reduce performance due to increased prompt length and the problem of LLMs overlooking relevant information. An exhaustive analysis of cases in which the reflection-based workflows were outperformed by the baseline approach showed that reflection tends to *over-reflect* on some queries, including objects that may not be related to the task at hand.

For future research, we plan to investigate the influence of additional object properties on the system's overall performance. A promising direction is to study how LLMs handle uncertainties in input data, such as those arising from Voxeland's object classification.

Author Contributions: Conceptualization, J.G.-J. and J.R.R.-S.; methodology, J.M.-R. and J.R.R.-S.; software, J.M.-R. and J.L.M.-B.; validation, J.M.-R.; formal analysis, J.M.-R. and J.R.R.-S.; investigation, J.M.-R. and J.R.R.-S.; resources, J.M.-R.; data curation, J.M.-R. and J.L.M.-B.; writing—original draft preparation, J.M.-R.; writing—review and editing, J.R.R.-S., J.L.M.-B. and J.G.-J.; visualization, J.M.-R. and J.R.R.-S.; supervision, J.R.R.-S. and J.G.-J.; project administration, J.R.R.-S. and J.G.-J.; funding acquisition, J.R.R.-S. and J.G.-J.. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been developed in the context of the project Voxeland (JA.B1-09) funded by the University of Málaga, and the projects ARPEGGIO (PID2020-117057GB-I00) and MINDMAPS (PID2023-148191NB-I00) funded by the Ministry of Science, Innovation and Universities of Spain.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The implemented code and the created dataset presented in the study are openly available in the GitHub repository <https://github.com/MAPIRlab/llm-robotics-reflection.git>.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

API	Application Programming Interface
GUI	Graphical User Interface
JSON	Javascript Object Notation
LLM	Large Language Model
LVLM	Large Vision-Language Model
OWL	Web Ontology Language
PDDL	Planning Domain Definition Language
RDF	Resource Description Framework
RGB-D	Red Green Blue - Depth
SfM	Structure from Motion
SLAM	Simultaneous Localization and Mapping
SPARQL	SPARQL Protocol and RDF Query Language
SWRL	Semantic Web Rule Language

Appendix A

The queries in the dataset are listed in Table A1. As explained in Section 4.3, queries are divided into three types based on the required reasoning: *Descriptive*, *Affordance*, and *Negation*. Each type is further subdivided into queries of varying difficulty.

Table A1. List of queries included in the dataset organized by type and difficulty.

Type	Difficulty	Query ID	Query
Descriptive	Easy	query_01	I'm searching for a bike in the room
		query_02	Where is the bag?
		query_03	Is there a refrigerator?
		query_04	I'm searching the bed
		query_05	I need to go to the toilet
	Difficult	query_06	I need the book that is on a table
		query_07	Is there a kitchen sink?
		query_08	A chair close to a table
		query_09	Is there a microwave that is close to an oven?
		query_10	A couch that is in front of the TV
Affordance	Easy	query_11	What can I use to hold my coffee one moment?
		query_12	I need to write notes during a meeting
		query_13	I would like to go out for some exercise
		query_14	Where can I leave the dirty dishes?
		query_15	I am tired, where can I sit?
	Difficult	query_16	Where can I sit and observe the entire view of the room without obstruction?
		query_17	What could show the up-to-the-minute weather forecast?
		query_18	I want to drop a delicate object, away from foot traffic
		query_19	I want to perform my oral hygiene routine without anyone seeing me
		query_20	What can I sit on that is near a book?
Negative	Easy	query_21	Tell me something to sit on that is not a chair
		query_22	Find somewhere to eat other than the table
		query_23	Tell me something I can sit on that is not a couch
		query_24	Tell me something I can open that is not a refrigerator
		query_25	Something that is not at room temperature
	Difficult	query_26	Something that emits sound but is not the TV, because it is turned off
		query_27	Identify an object that I can safely stand on to reach a high shelf, which is not a chair
		query_28	Find a storage item I can use that is not a backpack or handbag
		query_29	Locate an item that provides light but isn't a table lamp
		query_30	Something where I can deposit liquids, that doesn't close?

For convenience, a GUI was developed to annotate the ground-truth data (see Figure A1). As shown, it consists of two panels: one displaying the objects in the semantic map (left) and another containing the queries and the annotated responses (right). Each object in the semantic map is represented with its identifier (obj1, obj3, obj6, etc.), its most probable semantic class, and its bounding box coordinates and extension. Each query is also characterized by its identifier (query_01, query_02, query_03, etc.), the query text, and the list of annotated response objects. When a query is selected, the ground-truth objects are highlighted in the semantic map. All the information editable by the program is saved into JSON files in the background, which can be saved/loaded as needed.

Semantic map

Load semantic map JSON file

Loaded file: scannet_scene0673_04.json

Inspect scene!

Object	Most Probable Obj	BoundingBoxCenter	BoundingBoxSize	Results
obj1	remote (0.70)	3.3 2.0 0.37	0.74 1.18 0.51	remote: 86.61, dining table: 19.76, cup: 8.2
obj3	couch (0.74)	4.35 2.01 0.59	1.57 2.25 1.1	couch: 204.74, chair: 16.97, book: 16.64, la
obj6	bicycle (0.91)	5.61 3.85 0.53	0.85 0.66 0.94	bicycle: 31.54, clock: 2.32, chair: 0.65
obj7	bicycle (0.94)	5.5 3.15 0.56	0.42 0.6 0.73	bicycle: 23.13, chair: 1.35
obj8	chair (0.88)	4.58 3.69 0.48	2.31 2.07 0.96	chair: 103.91, bed: 4.09, couch: 2.94, persc
obj28	chair (0.43)	4.02 6.74 0.52	1.64 2.09 1.02	chair: 68.22, bed: 63.90, person: 6.44, couc
obj29	chair (0.52)	3.41 4.46 0.36	1.89 2.46 0.71	chair: 32.28, backpack: 6.35, bed: 4.88, sui
obj31	suitcase (0.75)	4.64 4.88 0.17	1.48 0.87 0.36	suitcase: 45.52, chair: 6.29, handbag: 5.54,
obj34	chair (0.70)	2.67 4.08 0.6	0.57 0.59 0.71	chair: 34.21, dining table: 14.45
obj36	donut (0.81)	2.28 3.56 0.77	0.18 0.19 0.66	donut: 32.94, remote: 6.18, spoon: 0.81, bi
obj40	book (0.89)	2.23 4.57 1.09	0.55 0.76 1.14	book: 65.48, chair: 5.27, bottle: 1.87, clock
obj33	tv (0.88)	1.85 2.65 1.19	0.49 1.15 0.74	tv: 54.08, chair: 6.45, dog: 0.67
obj64	backpack (0.76)	2.54 1.86 0.26	0.66 1.28 0.33	backpack: 21.02, person: 5.46, suitcase: 1.1
obj65	dining table (0.49)	3.14 1.02 0.6	0.56 0.56 0.43	dining table: 16.26, chair: 9.57, remote: 4.1
obj67	laptop (0.89)	3.66 0.78 0.36	0.43 0.46 0.17	laptop: 24.89, book: 2.93
obj70	bicycle (0.98)	4.97 0.51 0.64	0.6 0.58 0.63	bicycle: 30.88, person: 0.60
obj72	bicycle (1.00)	5.33 1.66 1.06	0.39 0.57 0.32	bicycle: 26.09
obj98	handbag (0.38)	6.25 6.97 0.44	3.31 2.11 0.81	handbag: 7.41, dining table: 4.83, backpack
obj99	bicycle (0.97)	7.47 5.38 0.7	0.74 1.91 1.0	bicycle: 41.08, clock: 1.27
obj132	sink (0.59)	2.27 8.2 1.24	1.72 0.69 0.47	sink: 43.95, bottle: 22.09, cup: 6.64, toothb
obj134	bottle (1.00)	0.55 7.32 1.43	0.2 0.24 0.21	bottle: 29.63
obj140	cup (0.57)	2.42 8.2 1.26	1.42 0.69 0.43	cup: 20.63, bottle: 14.06, sink: 0.76, dining
obj143	bottle (0.67)	2.42 8.2 1.26	1.42 0.69 0.43	bottle: 21.60, cup: 9.53, toothbrush: 0.56, i
obj149	oven (0.80)	2.93 6.89 0.77	0.87 1.01 1.23	oven: 27.62, refrigerator: 5.67, bowl: 0.63,

Ground truth

Load queries YAML file

Loaded ground truth JSON file

Loaded file: scannet_scene0673_04.json

Query ID	Query	Response
query_01	I'm searching for a bike in the room	obj72, obj70, obj99, obj7, obj6
query_02	Where is the bag?	obj31, obj64, obj98
query_03	Is there a refrigerator?	
query_04	I'm searching the bed	
query_05	I need to go to the toilet	
query_06	I need the book that is on a table	
query_07	Is there a kitchen sink?	obj132
query_08	A chair close to a table	obj34, obj28, obj29
query_09	Is there a microwave that is close to an oven?	
query_10	A couch that is in front of the TV	obj3
query_11	What can I use to hold my coffee one moment?	obj34, obj132
query_12	I need to write notes during a meeting	obj67, obj34
query_13	I would like to go out for some exercise	obj72, obj70, obj99, obj7, obj6
query_14	Where can I leave the dirty dishes?	obj132
query_15	I am tired, where can I sit?	obj3, obj8, obj34, obj28, obj29
query_16	Where can I sit and observe the entire view of the room with	obj8
query_17	What could show the up-to-the-minute weather forecast?	obj67, obj53
query_18	I want to drop a delicate object, away from foot traffic	obj28
query_19	I want to perform my oral hygiene routine without anyone s	obj132
query_20	What can I sit on that is near a book?	obj34
query_21	Tell me something to sit on that is not a chair	obj3
query_22	Find somewhere to eat other than the table	obj3, obj8, obj28, obj29
query_23	Tell me something I can sit on that is not a couch	
query_24	Tell me something I can open that is not a refrigerator	obj40, obj149, obj31, obj64
query_25	Something that is not at room temperature	obj149
query_26	Something that emits sound but is not the TV, because it is t	obj67, obj149
query_27	Identify an object that I can safely stand on to reach a high s	obj3, obj8
query_28	Find a storage item I can use that is not a backpack or handb	obj31

Clear

Figure A1. GUI developed to create the ground-truth data. The left panel shows the semantic map with its list of objects, while the right panel displays the queries and their annotated object responses.

References

- Rubio, F.; Valero, F.; Llopis-Albert, C. A review of mobile robots: Concepts, methods, theoretical framework, and applications. *International Journal of Advanced Robotic Systems* **2019**, *16*.
- Galindo, C.; Saffiotti, A.; Coradeschi, S.; Buschka, P.; Fernandez-Madriral, J.A.; González, J. Multi-hierarchical semantic maps for mobile robotics. In Proceedings of the 2005 IEEE/RSJ international conference on intelligent robots and systems. IEEE, 2005, pp. 2278–2283.
- Nüchter, A.; Wulf, O.; Lingemann, K.; Hertzberg, J.; Wagner, B.; Surmann, H. 3D mapping with semantic knowledge. In Proceedings of the RoboCup 2005: Robot Soccer World Cup IX 9. Springer, 2006, pp. 335–346.
- Ranganathan, A.; Dellaert, F. Semantic Modeling of Places using Objects. In Proceedings of the Robotics: Science and Systems, 2007, Vol. 3, pp. 27–30.
- Meger, D.; Forssén, P.E.; Lai, K.; Helmer, S.; McCann, S.; Southey, T.; Baumann, M.; Little, J.J.; Lowe, D.G. Curious george: An attentive semantic robot. *Robotics and Autonomous Systems* **2008**, *56*, 503–511.
- Uschold, M.; Gruninger, M. Ontologies: Principles, methods and applications. *The knowledge engineering review* **1996**, *11*, 93–136.
- Lassila, O. Resource Description Framework (RDF) Model and Syntax **1997**.
- McGuinness, D.L. OWL Web Ontology Language Overview. *W3C Member Submission* **2004**.
- Prud'hommeaux, E.; Seaborne, A. SPARQL Query Language for RDF. <https://www.w3.org/TR/rdf-sparql-query/>, 2008. W3C Recommendation.
- Horrocks, I.; Patel-Schneider, P.F.; Boley, H.; Tabet, S.; Grosz, B.; Dean, M.; et al. SWRL: A semantic web rule language combining OWL and RuleML. *W3C Member submission* **2004**, *21*, 1–31.
- Silver, T.; Hariprasad, V.; Shuttleworth, R.S.; Kumar, N.; Lozano-Pérez, T.; Kaelbling, L.P. PDDL Planning with Pretrained Large Language Models. In Proceedings of the NeurIPS 2022 Foundation Models for Decision Making Workshop, 2022.
- Vaswani, A. Attention is all you need. *Advances in Neural Information Processing Systems* **2017**.
- Radford, A. Improving language understanding by generative pre-training **2018**.
- Devlin, J.; Chang, M.W.; Lee, K.; Toutanova, K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, 2019, [arXiv:cs.CL/1810.04805].
- Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I.; et al. Language models are unsupervised multitask learners. *OpenAI blog* **2019**, *1*, 9.
- OpenAI. GPT-4 Technical Report. <https://openai.com/research/gpt-4>, 2023. Accessed: 2024-12-05.

17. Gu, Q.; Kuwajerwala, A.; Morin, S.; Jatavallabhula, K.M.; Sen, B.; Agarwal, A.; Rivera, C.; Paul, W.; Ellis, K.; Chellappa, R.; et al. ConceptGraphs: Open-Vocabulary 3D Scene Graphs for Perception and Planning, 2023, [arXiv:cs.RO/2309.16650].
18. Rana, K.; Haviland, J.; Garg, S.; Abou-Chakra, J.; Reid, I.; Suenderhauf, N. SayPlan: Grounding Large Language Models using 3D Scene Graphs for Scalable Robot Task Planning, 2023, [arXiv:cs.RO/2307.06135].
19. Gao, T.; Fisch, A.; Chen, D. Making Pre-trained Language Models Better Few-shot Learners, 2021, [arXiv:cs.CL/2012.15723].
20. Ji, Z.; Lee, N.; Frieske, R.; Yu, T.; Su, D.; Xu, Y.; Ishii, E.; Bang, Y.J.; Madotto, A.; Fung, P. Survey of hallucination in natural language generation. *ACM Computing Surveys* **2023**, *55*, 1–38.
21. Peng, B.; Galley, M.; He, P.; Cheng, H.; Xie, Y.; Hu, Y.; Huang, Q.; Liden, L.; Yu, Z.; Chen, W.; et al. Check Your Facts and Try Again: Improving Large Language Models with External Knowledge and Automated Feedback, 2023, [arXiv:cs.CL/2302.12813].
22. Patil, S.G.; Zhang, T.; Wang, X.; Gonzalez, J.E. Gorilla: Large Language Model Connected with Massive APIs, 2023, [arXiv:cs.CL/2305.15334].
23. Schick, T.; Dwivedi-Yu, J.; Dessi, R.; Raileanu, R.; Lomeli, M.; Zettlemoyer, L.; Cancedda, N.; Scialom, T. Toolformer: Language Models Can Teach Themselves to Use Tools, 2023, [arXiv:cs.CL/2302.04761].
24. Madaan, A.; Tandon, N.; Gupta, P.; Hallinan, S.; Gao, L.; Wiegrefe, S.; Alon, U.; Dziri, N.; Prabhmaye, S.; Yang, Y.; et al. Self-Refine: Iterative Refinement with Self-Feedback, 2023, [arXiv:cs.CL/2303.17651].
25. Shinn, N.; Cassano, F.; Berman, E.; Gopinath, A.; Narasimhan, K.; Yao, S. Reflexion: Language Agents with Verbal Reinforcement Learning, 2023, [arXiv:cs.AI/2303.11366].
26. Dai, A.; Chang, A.X.; Savva, M.; Halber, M.; Funkhouser, T.; Nießner, M. ScanNet: Richly-annotated 3D Reconstructions of Indoor Scenes. In *Proceedings of the Proc. Computer Vision and Pattern Recognition (CVPR)*, IEEE, 2017.
27. Hua, B.S.; Pham, Q.H.; Nguyen, D.T.; Tran, M.K.; Yu, L.F.; Yeung, S.K. SceneNN: A Scene Meshes Dataset with aNNotations. In *Proceedings of the 2016 Fourth International Conference on 3D Vision (3DV)*, 2016, pp. 92–101. <https://doi.org/10.1109/3DV.2016.18>.
28. Matez-Bandera, J.L.; Ojeda, P.; Monroy, J.; Gonzalez-Jimenez, J.; Ruiz-Sarmiento, J.R. Voxeland: Probabilistic Instance-Aware Semantic Mapping with Evidence-based Uncertainty Quantification, 2024, [arXiv:cs.RO/2411.08727].
29. Galindo, C.; Fernandez-Madrigal, J.A.; Gonzalez, J. Multihierarchical Interactive Task Planning: Application to Mobile Robotics. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* **2008**, *38*, 785–798. <https://doi.org/10.1109/TSMCB.2008.920227>.
30. Ruiz-Sarmiento, J.R.; Galindo, C.; Gonzalez-Jimenez, J. Building multiversal semantic maps for mobile robot operation. *Knowledge-Based Systems* **2017**, *119*, 257–272. <https://doi.org/https://doi.org/10.1016/j.knosys.2016.12.016>.
31. Sünderhauf, N.; Pham, T.T.; Latif, Y.; Milford, M.; Reid, I. Meaningful Maps With Object-Oriented Semantic Mapping, 2017, [arXiv:cs.RO/1609.07849].
32. McCormac, J.; Handa, A.; Davison, A.; Leutenegger, S. SemanticFusion: Dense 3D Semantic Mapping with Convolutional Neural Networks, 2016, [1609.05130]. <https://doi.org/https://doi.org/10.48550/arXiv.1609.05130>.
33. Rünz, M.; Buffier, M.; Agapito, L. MaskFusion: Real-Time Recognition, Tracking and Reconstruction of Multiple Moving Objects, 2018, [1804.09194]. <https://doi.org/https://doi.org/10.48550/arXiv.1804.09194>.
34. Qian, J.; Chatrath, V.; Yang, J.; Servos, J.; Schoellig, A.P.; Waslander, S.L. POCD: Probabilistic Object-Level Change Detection and Volumetric Mapping in Semi-Static Scenes, 2022, [2205.01202]. <https://doi.org/https://doi.org/10.48550/arXiv.2205.01202>.
35. Qian, J.; Chatrath, V.; Servos, J.; Mavrinnac, A.; Burgard, W.; Waslander, S.L.; Schoellig, A.P. POV-SLAM: Probabilistic Object-Aware Variational SLAM in Semi-Static Environments, 2023, [2307.00488].
36. Ren, S. Faster r-cnn: Towards real-time object detection with region proposal networks. *arXiv preprint arXiv:1506.01497* **2015**.
37. He, K.; Gkioxari, G.; Dollár, P.; Girshick, R. Mask r-cnn. In *Proceedings of the Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969. <https://doi.org/https://doi.org/10.48550/arXiv.1703.06870>.
38. Ahn, M.; Brohan, A.; Brown, N.; Chebotar, Y.; Cortes, O.; David, B.; Finn, C.; Fu, C.; Gopalakrishnan, K.; Hausman, K.; et al. Do As I Can, Not As I Say: Grounding Language in Robotic Affordances, 2022, [arXiv:cs.RO/2204.01691].

39. Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; Stone, P. LLM+P: Empowering Large Language Models with Optimal Planning Proficiency, 2023, [arXiv:cs.AI/2304.11477].
40. Huang, W.; Xia, F.; Xiao, T.; Chan, H.; Liang, J.; Florence, P.; Zeng, A.; Tompson, J.; Mordatch, I.; Chebotar, Y.; et al. Inner Monologue: Embodied Reasoning through Planning with Language Models, 2022, [arXiv:cs.RO/2207.05608].
41. Ouyang, L.; Wu, J.; Jiang, X.; Almeida, D.; Wainwright, C.L.; Mishkin, P.; Zhang, C.; Agarwal, S.; Slama, K.; Ray, A.; et al. Training language models to follow instructions with human feedback, 2022, [arXiv:cs.CL/2203.02155].
42. Madaan, A.; Tandon, N.; Rajagopal, D.; Clark, P.; Yang, Y.; Hovy, E. Think about it! Improving defeasible reasoning by first modeling the question scenario. In Proceedings of the Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing; Moens, M.F.; Huang, X.; Specia, L.; Yih, S.W.t., Eds., Online and Punta Cana, Dominican Republic, 2021; pp. 6291–6310. <https://doi.org/10.18653/v1/2021.emnlp-main.508>.
43. Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Ichter, B.; Xia, F.; Chi, E.; Le, Q.; Zhou, D. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models, 2023, [arXiv:cs.CL/2201.11903].
44. Chang, K.; Xu, S.; Wang, C.; Luo, Y.; Xiao, T.; Zhu, J. Efficient Prompting Methods for Large Language Models: A Survey, 2024, [arXiv:cs.CL/2404.01077].
45. Yang, K.; Tian, Y.; Peng, N.; Klein, D. Re3: Generating Longer Stories With Recursive Reprompting and Revision, 2022, [arXiv:cs.CL/2210.06774].
46. Qian, C.; Liu, W.; Liu, H.; Chen, N.; Dang, Y.; Li, J.; Yang, C.; Chen, W.; Su, Y.; Cong, X.; et al. ChatDev: Communicative Agents for Software Development, 2024, [arXiv:cs.SE/2307.07924].
47. Hong, S.; Zhuge, M.; Chen, J.; Zheng, X.; Cheng, Y.; Zhang, C.; Wang, J.; Wang, Z.; Yau, S.K.S.; Lin, Z.; et al. MetaGPT: Meta Programming for A Multi-Agent Collaborative Framework, 2024, [arXiv:cs.AI/2308.00352].
48. Wu, Q.; Bansal, G.; Zhang, J.; Wu, Y.; Li, B.; Zhu, E.; Jiang, L.; Zhang, X.; Zhang, S.; Liu, J.; et al. AutoGen: Enabling Next-Gen LLM Applications via Multi-Agent Conversation, 2023, [arXiv:cs.AI/2308.08155].
49. Brown, T.B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; et al. Language Models are Few-Shot Learners, 2020, [arXiv:cs.CL/2005.14165].
50. Bender, E.M.; Gebru, T.; McMillan-Major, A.; Shmitchell, S. On the Dangers of Stochastic Parrots: Can Language Models Be Too Big? In Proceedings of the Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency, New York, NY, USA, 2021; FAccT '21, p. 610–623. <https://doi.org/10.1145/3442188.3445922>.
51. Matez-Bandera, J.L.; Monroy, J.; Gonzalez-Jimenez, J. Sigma-FP: Robot Mapping of 3D Floor Plans With an RGB-D Camera Under Uncertainty. *IEEE Robotics and Automation Letters* **2022**, *7*, 12539–12546. <https://doi.org/10.1109/LRA.2022.3220156>.
52. Morilla-Cabello, D.; Mur-Labadia, L.; Martinez-Cantin, R.; Montijano, E. Robust Fusion for Bayesian Semantic Mapping. In Proceedings of the 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2023, pp. 76–81. <https://doi.org/10.1109/IROS55552.2023.10342253>.
53. Jang, J.; Ye, S.; Seo, M. Can Large Language Models Truly Understand Prompts? A Case Study with Negated Prompts, 2022, [arXiv:cs.CL/2209.12711].
54. Team, G.; Anil, R.; Borgeaud, S.; Alayrac, J.B.; Yu, J.; Soricut, R.; Schalkwyk, J.; Dai, A.M.; Hauth, A.; Millican, K.; et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* **2023**.
55. Liu, N.F.; Lin, K.; Hewitt, J.; Paranjape, A.; Bevilacqua, M.; Petroni, F.; Liang, P. Lost in the Middle: How Language Models Use Long Contexts, 2023, [arXiv:cs.CL/2307.03172].

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.