LONG-TERM CHANGING GROUNDING: 3D VISUAL GROUNDING IN CONTINUOUSLY VARYING SCENES

Anonymous authors
Paper under double-blind review

ABSTRACT

Most existing 3D visual grounding tasks are conducted in static scenes that capture the scene layout only at a single instant. However, in real-world scenarios, object arrangements change over time. It is impractical to rescan the scene every time an object moves during 3D visual grounding. To address this issue, we propose a novel task, 3D long-term changing grounding, and construct a new dataset named TemScan. In the TemScan dataset, object layouts change dynamically over time, requiring models to balance perception costs while accurately comprehending the scene. Furthermore,we propose a detect-then-ground approach to address this task. Specifically, we first fine-tune a pre-trained 3D convolutional network on the ScanNet dataset to predict 3D object proposals, then build structured representations of dynamic scenes via an uncertainty-aware scene graph, and finally introduce the plan-act-evaluate reasoning module to predict the target object. In addition,we leverage LLM to distill compact and reusable skill primitives, reducing token consumption by 43% and inference latency by 18%.Experimental results on the TemScan dataset show that our method achieves accuracies of 34.98% and 34.51% under the Acc@0.25 and Acc@0.5 metrics, respectively.

1 INTRODUCTION

3D Visual Grounding (3DVG) constitutes a pivotal yet challenging task at the intersection of 3D computer vision and multimodal intelligence. It aims to precisely localise and bound a target object within a 3D scene, conditioned on a natural language description.Recently, Large Language Models (LLMs) and Vision-Language Models (VLMs) have been increasingly integrated into this domain(Jin et al., 2026; Li et al., 2025b; Xu et al., 2024; Yang et al., 2024; Yuan et al., 2024). By harnessing the formidable reasoning capacities of LLMs, recent methods achieve superior and more robust training-free or open-vocabulary 3D spatial reasoning.

Current 3DVG benchmarks, including ScanRefer(Chen et al., 2020), ReferIt3D(Achlioptas et al., 2020), and Multi3DRefer(Zhang et al., 2023), are predominantly constructed based on static scenes with fixed object layouts. In contrast, real-world scenarios are inherently dynamic, where objects are continuously added, moved, or removed over time. Such static scene constraints render existing grounding methods suboptimal and impractical for dynamic real-world environments. Although existing datasets such as 3RScan(Wald et al., 2019) consider scene variations,some methods based on this dataset, such as ChangGrounding(Hu et al., 2025), take into account the problem of object grounding under scene object changes.They only provide discrete, non-sequential scene pairs without continuous scene changes. However, these methods only consider multiple scene distributions at a specific time relative to the original scene, and do not address the 3DVG task under continuous scene changes. And in real-world applications, however, the layout of objects in a scene changes continuously over time. Given the complexity and diversity of objects in a scene, predicting such changes is difficult and costly. Moreover, rescanning the scene in real time to enable the model to understand the scene is also impractical.

To bridge this gap, we introduce a novel task named 3D long-term changing grounding (3DLG). Formally, this task aims to localize dynamic target objects within varying 3D scenes at multiple timestamps according to textual guidance. Since no existing benchmark supports the study of 3DLG, we construct a new dataset for this task named TemScan. To build TemScan, we leverage ProcTHOR(Deitke et al., 2022) within AI2-THOR(Kolve et al., 2017) to programmatically generate a

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

set of indoor scenes corresponding to the initial timestamp t_0. Subsequent timestamps are populated by sequentially applying random object edits (addition, movement, or removal) to scenes from the prior timestamp, guided by a sliding-window collision detection strategy. Point clouds are sampled from scene mesh surfaces along with instance-level 3D bounding box annotations. For language annotations, we adopt Sr3D-style templates to automatically generate text queries according to inter-object spatial relationships. Excluding t_0, each timestamp contains a balanced amount of queries targeting unchanged and changed objects, facilitating explicit evaluation of grounding performance amid dynamic scene variations.

We also develop a detect-then-ground approach consisting of point cloud target prediction, uncertainty-aware scene graph modeling, and agent-based scene reasoning and grounding.For point cloud object prediction, we initialize our backbone with the FCAF3D(Rukhovich et al., 2022) pre-trained on ScanNet(Dai et al., 2017) point clouds and fine-tune the network on our TemScan dataset. Different from static 3D grounding, objects in real-world scenes continuously change over time, making historical scene memory progressively unreliable. To reduce unnecessary scene rescanning while faithfully capturing the real-time object distribution, we build an uncertainty-aware scene graph to structurally organize intra-scene object information across timestamps. Each object in the graph is assigned an uncertainty score that quantifies its temporal unreliability. Before processing each query, our method selectively rescans the scene room with the highest uncertainty and resets the uncertainty scores of updated objects to zero. In contrast, the uncertainty scores of unupdated objects increase as time elapses. Moreover, we implement an agent-based plan-act-evaluate Reasoning Module to perform long-term changing grounding reasoning. The agent adopts a three-stage plan-act-evaluate architecture, where the plan node conducts reasoning analysis and tool scheduling, the act node executes specific scene-interaction tools, and the evaluate node diagnoses reasoning flaws and corrects potential errors. In addition, we summarize effective reasoning trajectories from LLM inference processes to extract reusable agent skills. Equipped with these distilled skills, our agent significantly reduces token consumption and inference latency while preserving overall grounding accuracy.

Experimental results show that the agent can achieve an accuracy of 34.98% under the Acc@0.25 metric and 34.51% under the Acc@0.5 metric on the TemScan. Regarding the cost, the average scanned area per query is 9.19 m^2, and the average latency is 10.20 seconds. Compared with alternative strategies, this approach successfully balances both accuracy and cost.

In summary, our main contributions are as follows:

- We propose a new task, 3D long-term changing grounding (3DLG), which aims to achieve accurate localization under dynamic scene while minimizing computational and token costs.And we also build a 3D long-term changing grounding dataset, TemScan, by programmatically generating pointclouds and textual queries.
- We propose a detect-then-ground approach for 3DLG, introducing an uncertainty-aware scene graph and building a Plan-Act-Evaluate Reasoning agent with affordable time and token consumption costs.
- We conduct extensive experiments on the TemScan dataset, achieving 34.98% and 34.51% accuracy under the Acc@0.25 and Acc@0.5 metrics, respectively.

2 Related Works

SeeGround(Li et al., 2025a) introduced a zero-shot 3D visual grounding framework, SeeGround, leveraging pre-trained 2D foundation models to bypass 3D annotation training. It designed viewpoint-adaptive and fusion-alignment modules to generate query-matched rendered images coupled with spatial text, incorporating fine-grained visual clues such as texture and orientation. SPAZER(Jin et al., 2026) proposed a spatial-semantic progressive reasoning agent, addressing the limitations of existing methods that lean heavily toward a single 2D or 3D modality. It first conducts global multi-view rendering to screen candidate objects, and then maps them back to real 2D images for fine-grained semantic verification, achieving joint 3D-2D decision-making.SeqVLM(Lin et al., 2025) adopts a proposal-guided multi-view projection strategy, which first filters semantically matched 3D candidates, and then maps them to real scene images to construct concatenated sequences, cooperating with iterative VLM reasoning to mitigate computational overhead.

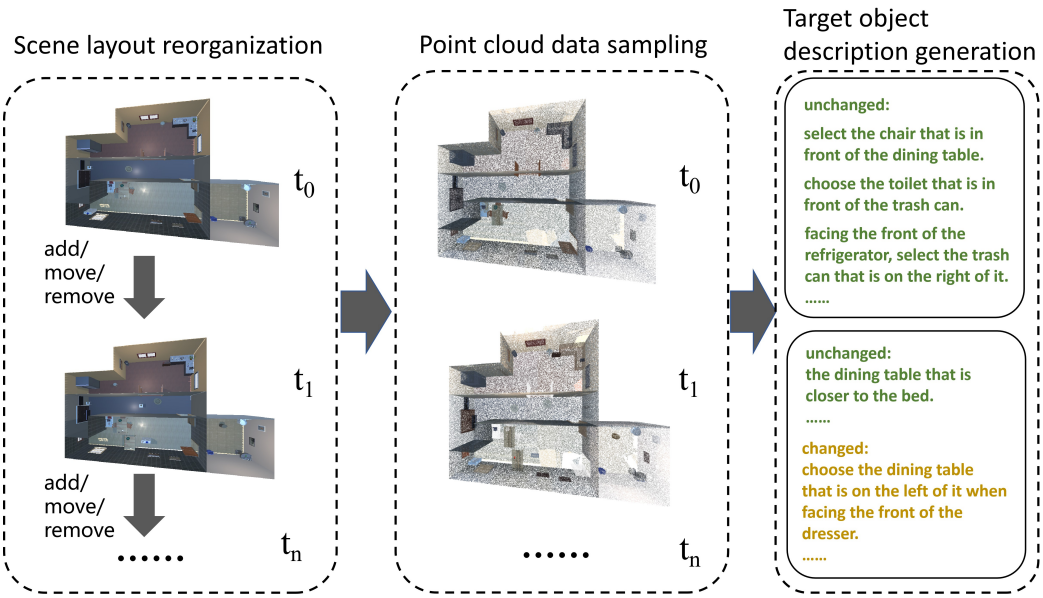

Figure 1: The Construction Process of the 3D Long-Term Changing Grounding Dataset TemScan.

3RScan(Wald et al., 2019) introduced the 3D object instance relocalization task, aiming to estimate the 6DoF poses of objects in dynamic indoor environments using RGB-D scans captured at different timestamps. They released the 3RScan dataset and benchmark comprising 1,482 scans, and proposed a data-driven approach based on a geometric multi-scale neural network. Meanwhile, ChangingGrounding(Hu et al., 2025) proposed a memory-driven 3D visual grounding task and the ChangingGrounding benchmark within dynamic scenes. Leveraging 3RScan, they constructed a dataset with 267k queries and developed the Mem-ChangingGrounder framework, which blends historical memory with multi-view scans to balance localization accuracy against robot scene exploration overhead. Although these tasks account for the changing nature of scenes, they lack continuity, where each modified scene merely constitutes an isolated pair relative to the original scene.Although these works consider the 3DVG task under scene object changes, the data they use are mostly discrete, focusing on scene changes at an arbitrary unknown time relative to a certain reference time, without a complete temporal workflow.

3 METHODOLOGY

3.1 THE TEMSCAN DATASET

3.1.1 DATASET CONSTRUCTION

As illustrated in Figure 1, we first programmatically generate an indoor scene at time t_0 via ProcTHOR(Deitke et al., 2022), following the standard AI2-THOR(Kolve et al., 2017) format. Starting from this initial scene, we perform d sequential object modifications. Each modification randomly executes one of three operations—*add*, *move*, or *remove*—with equal probability: adding a new object into the scene, relocating an existing object, or deleting an object.

Starting from the initial indoor scene at timestamp t_0, we perform a series of object modifications. When carrying out *add* and *move* operations, we employ a sliding-window collision detection algorithm. Concretely, the room floor is discretized into an n-meter grid, onto which the top-down projection of each object is mapped. Grid cells occupied by objects are marked as 1, and empty cells are marked as 0. When adding or moving an object, all grid cells covered by its projection must equal 0; otherwise, a collision is detected. The sliding window starts traversal from a randomly sampled grid cell and continues until a collision-free placement is found, which indicates a successful modification. If collisions arise at every candidate position after traversing all available grid cells, we replace the object with an alternative one featuring a smaller projected footprint and

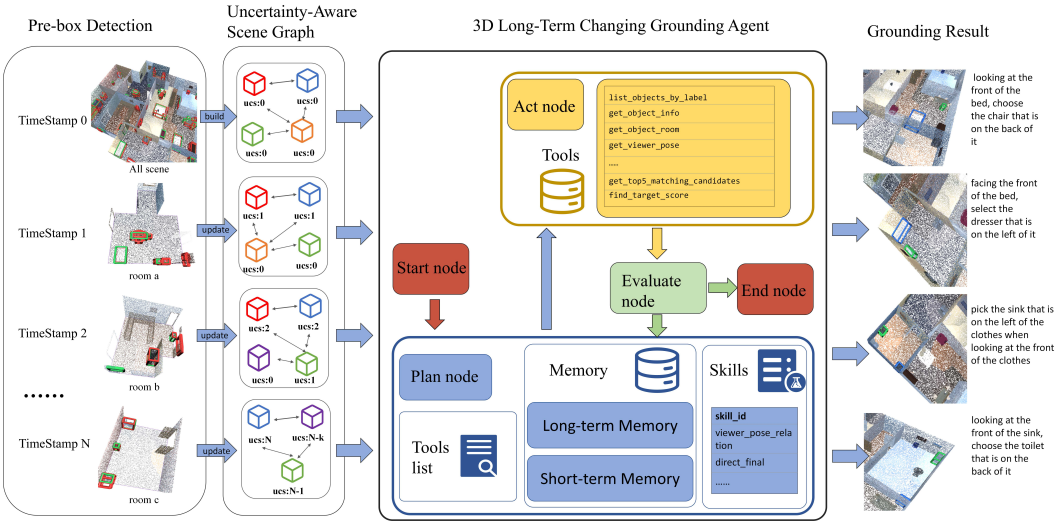

Figure 2: The pipeline of the entire 3D long-term changing grounding framework. In the uncertainty-aware scene graph shown in the figure, usc stands for uncertainty score.

repeat the placement attempt, capped at m retries. This procedure yields the scene at timestamp t_1. Repeating the same workflow, we derive all subsequent scenes from t_2 to t_7 by iteratively modifying the scene from the preceding timestamp.

Subsequently, we programmatically generate textual queries by leveraging the templates of Sr3D from ReferIt3D(Achlioptas et al., 2020), where each timestamp contains d queries. If an object is added to the scene or moved at the current timestamp, it is classified into the changed category; otherwise, it is assigned to the unchanged category. For the d queries at time t_0, all belong to the unchanged category. From time t_1 to t_7, the queries at each timestamp consist of $d/2$ unchanged and $d/2$ changed queries, aiming to ensure that the queries cover both moved and unmoved objects as comprehensively as possible.Then we obtain the scene 3D mesh via AI2-THOR and uniformly sample its surface to generate colored point clouds. This dataset also includes the object instance segmentation of the point cloud and the ground truth bounding box calculated for each instance.

3.1.2 Dataset Statistics

The dataset comprises a total of 200 scenes, each spanning 8 timestamps, which yields 1600 distinct point cloud scenes. On average, each scene contains 4 rooms, 32 object categories, and 1.35 million points. Due to the fact that some scenes contain fewer than 10 objects, the total number of generated queries is 14440.The number of queries d contained in each timestamp is set to 10, including 5 queries for unchanged objects and 5 queries for changed objects. The spatial relations between target and anchor objects within the queries are classified into three categories: allocentric (e.g., left, right, front, etc.), horizontal (e.g., closest, farthest, etc.), and support (e.g., above, below, supporting, etc.). Among all queries, 7997 belong to the unchanged category, while 6443 fall under the changed category, with an average of 8.74 edits per scene. Regarding the specific operations, *add* was executed 4714 times, *move* 4562 times, and *remove* 4724 times. The most frequent target object categories include bed (2231), chair (1726), tv (1157), dresser (1114), and dining table (1091).For more detailed statistical information, please refer to the appendix.

3.2 3D Long-Term Changing Grounding Framework

3.2.1 Point Cloud Instance Detection

We first explored generating 3D bounding boxes by back-projecting 2D object masks into 3D space. The masks and object categories were obtained from 2D images via SAM(Kirillov et al., 2023) and DINO(Ren et al., 2024). Unfortunately, this pipeline brought substantial noise to the extracted instances and led to unsatisfactory detection quality. For this reason, we adopt the fully supervised

FCAF3D(Rukhovich et al., 2022) model to produce predicted bounding boxes for grounding evaluation. We initialize FCAF3D with weights pre-trained on the ScanNet(Dai et al., 2017) point cloud dataset while keeping the backbone and neck unchanged. To match the 32 object categories in our dataset, we adjust the classification head from 18 to 32 output dimensions and reinitialize it. We then fine-tune the model on our 3D Long-Term Changing Grounding dataset and obtain reliable object detection results. Further quantitative analysis of detection performance is provided in the experiments section.

3.2.2 UNCERTAINTY-AWARE SCENE GRAPH

In 3D long-term changing grounding, objects within the scene shift and change over time, which may render historical scene memories inaccurate as time elapses. Accordingly, how to reduce redundant scene scanning while faithfully capturing the real object distribution constitutes a key open challenge. To tackle this issue, we build a 3D uncertainty-aware scene graph that dynamically refreshes internal information during 3D visual grounding. Each node corresponds to one scene object and stores its predicted bounding box coordinates, category label, affiliated room, and uncertainty score, while edges are defined as coordinate difference vectors between pairs of objects. A higher node uncertainty score implies a larger likelihood that the associated object has moved away from its previous position. All uncertainty scores are initialized to 0 at t_0 and increase by 1 at each subsequent timestamp. Before processing each new query, the scene graph identifies the room with the highest average uncertainty score for re-scanning. It then updates the information inside this room and resets the corresponding uncertainty scores to 0. Equipped with this re-scanning strategy, the scene graph dynamically sustains accurate and complete scene memory throughout the entire grounding process.

3.2.3 PLAN-ACT-EVALUATE REASONING MODULE

As illustrated by the complete agent workflow in Figure 2,the agent is implemented via the LangGraph (LangChain AI, 2024) framework and comprises three key nodes: the plan node, the act node, and the evaluate node. Concretely, the plan node reasons to generate a list of tool calls; the act node executes these tools and returns corresponding outputs; the evaluate node audits reasoning outputs from the plan node. It detects logical flaws in the planning procedure. When severe errors or omissions are found, the evaluate node offers revision suggestions and triggers replanning. If it receives the "final" signal from the plan node instead, the evaluate node consolidates dialogue context and generates the final grounding prediction.

Before planning, the plan node can access the available tool set, as well as short-term and long-term memory buffers. Short-term memory stores historical reasoning traces for the ongoing query and prior feedback from the evaluate node, enabling the plan node to track the current stage and progress of grounding. Long-term memory corresponds to the persistent uncertainty-aware scene graph maintained across timestamps, which the plan node can query via tool functions to support decision-making.

Experimental results reveal that during the grounding process, the LLM consumes an exceptionally large number of tokens, and some of the reasoning processes are repetitive. To address this, we statistically analyze all tool invocation flows that achieve a localization accuracy higher than $\mathrm{Acc@0.5}$, and select the top-10 most frequent tool flows as the candidate skills. The semantic meaning of each tool within the flow and k randomly selected successful queries are fed into the LLM to generate a textual description for each skill. Subsequently, we extract the reasoning processes from k successful queries along with k failed instances for each skill. The LLM is then employed to summarize the positive and negative contexts, explicitly outputting when to utilize or bypass the skill. Therefore, at the very beginning of the reasoning process, the Plan node can inspect these skill descriptions to selectively invoke appropriate skills, thereby optimizing reasoning costs, reducing token consumption, and accelerating inference time.

For the toolset available for LLM invocation, we construct four major categories: object retrieval, spatial reasoning, scene scanning, and diagnosis and recovery. The Plan node selectively invokes these tools by inspecting their functional descriptions. For detailed tool definitions and core parameter specifications, please refer to the appendix.

Table 1: Object detection and localization performance. We present the overall performance alongside the top 6 and bottom 6 classes sorted by their AP@0.5 metric. Ratio indicates the percentage of ground-truth instances for each category. All values are reported in percentages (%).

Top 6 Classes (sorted by AP@0.5)				**Bottom 6 Classes** (sorted by AP@0.5)			
Class	Ratio (%)	AP@0.5	AR@0.5	Class	Ratio (%)	AP@0.5	AR@0.5
laundry hamper	0.14	100.00	100.00	safe	0.24	28.57	63.64
trash can	0.12	95.02	100.00	vacuum	0.08	20.35	42.86
cart	1.20	94.41	97.27	plunger	1.44	11.52	31.06
dresser	5.92	93.97	99.63	baseballbat	0.59	0.21	18.52
sofa	3.93	91.90	100.00	decorativebox	0.28	0.02	3.85
sink	3.94	90.09	99.72	ottoman	0.07	0.00	0.00
Overall	100.00	65.33	83.84				

During the grounding process of the agent, the LLM performs a comprehensive judgment to localize objects by comparing the spatial relations between pairs of objects in the scene graph, the coordinate values between the two objects, the object labels against the target object name in the query, as well as a rigidly calculated $total_score$. This manner of combining multiple types of information ensures that the LLM makes the best possible decisions as much as possible.

4 Experiments

4.1 Implementation Details

During the dataset construction, the grid size n for collision detection is set to 0.25 m, and the maximum number of collision detection attempts m is capped at 20. For FCAF3D training, 140 scenes are utilized as the training set,30 scenes as the validation set, and 30 scenes as the test set. Subsequent LLM inference experiments are conducted on these 30 test scenes. The voxel size for FCAF3D training is set to 0.01, and each point cloud scene is downsampled to $50,000$ points. The AdamW optimizer is employed with an initial learning rate of 0.001 and a weight decay of 1×10^{-4}. The model achieves its optimal performance on the validation set after training for 20 epochs on a single NVIDIA RTX 4090 GPU. During training, data augmentation is applied by horizontally and vertically flipping the point clouds. Predicted bounding boxes with a confidence score lower than 0.05 are filtered out and excluded from the scene graph construction.

During the agent inference process, the maximum loop cycle is limited to 6. The dialogue history inputted into the LLM is truncated to a maximum of the preceding 4 rounds. We employ CLIP ViT-B/32 as the text embedding generation model. For LLM inference, DeepSeek-Chat is utilized with a configuration consisting of a temperature of 0.0, a max tokens limit of 1024, a timeout of 90 seconds, and 3 retries. When generating skills, the number of randomly sampled instances k is chosen as 20.

4.2 Quantitative Results

As shown in Table 1, the model achieves an overall performance of 65.33% in AP@0.5 and 83.84% in AR@0.5 on the object detection task, demonstrating its strong recall capability. The table also lists the top-6 and bottom-6 object categories under the AP@0.5 metric. Category-wise analysis reveals that common objects, including cart, dresser, sofa, and sink, exhibit robust performance, which indicates that the model can accurately recognize categories with stable texture and geometric features. Conversely, the APs for categories such as ottoman (0.07%), decorativebox (0.28%), and baseballbat (0.59%) are notably low, reflecting the inherent challenges that small physical sizes and long-tailed categories pose to 3D detection models.

Table 2 compares the experimental results under different decision strategies. To investigate the effectiveness of the decision-making and grounding component of our method, experiments (1), (2), and (3) present the performance of traditional 3DVG methods on the TemScan dataset.We adopt GPT-4o as the LLM for all these methods. Before localizing each query, the entire scene is res-

Table 2: Performance comparison across different methods on multiple metrics.

Method	Acc@0.25 (%)	Acc@0.50 (%)	Scan Area (m^2/q)	Calls (/q)	Latency (s/q)
(1) seeground	34.18	33.90	87.35	2.00	11.73
(2) SeqVLM	33.20	32.93	87.35	2.15	13.14
(3) SPAZER	32.31	32.03	87.35	3.52	23.50
(4) plan-act-evaluate Agent (with gtbox)	52.06	50.47	8.21	2.38	8.37
(5) plan-act-evaluate Agent (our)	**34.98**	**34.51**	**9.19**	**2.93**	**10.20**
(6) React Agent	33.61	33.12	9.27	3.06	10.29
(7) Best Match	25.92	25.61	12.60	-	-
(8) Random	6.43	6.57	36.50	-	-

canned, aiming to accurately and faithfully reflect the real scene information. It can be observed that, given the true scene layout as input, our method still achieves leading performance compared with 3DVG methods on the Acc@0.25 and Acc@0.5 metrics, demonstrating the effectiveness of the decision-making component of our method. In addition, our model also has certain advantages in terms of time latency and scene scanning cost: while achieving leading localization accuracy, its scanning cost is only 10.53% of that of these methods. Experiment (4) reports the results obtained using ground-truth boxes, aiming to investigate the upper bound that the decision-making component can reach after excluding the interference of object bounding box prediction. It can be seen that it achieves an accuracy of 52.06% on this metric, proving the decision-making effectiveness of our method. Experiments (6), (7), and (8) adopt different decision-making logics. Experiment (6) is an agent framework that first reasons and then acts, experiment (7) uses the maximum score computed by functions as the final grounding result, and experiment (8) randomly selects objects in the scene graph as the grounding result. Through experimental comparison, it can be seen that our method, which first plans, then acts, and finally evaluates, achieves a 1.39% improvement on the Acc@0.5 metric over the single ReAct structure. Moreover, in terms of scanned area, the number of LLM calls, and time latency, our method is faster and incurs lower scene scanning costs.

Table 3: Ablation study on the key components of our proposed framework.

	Uncertainty-Aware Scene Graph	Plan-Act	Eval	Skills	Acc@0.25 (%)	Acc@0.50 (%)	Scan Area (m^2/q)	Tokens (/q)	Calls (/q)	Latency (s/q)
(1)	✓	✓	✓	✓	34.98	34.51	**9.19**	**13,336**	**2.93**	**10.20**
(2)	✓	✓	✓		**34.17**	**34.82**	9.24	23,025	3.75	12.43
(3)	✓	✓			31.93	31.46	9.37	19,391	3.05	11.27
(4)	✓				25.92	25.61	12.60	-	-	-
(5)		✓	✓	✓	16.31	15.72	87.35	14,216	3.01	10.58

As shown in Table 3, we also conduct ablation experiments on each module. Comparing experiments (1) and (2), we can observe the role of skills in reducing costs. After applying skills, the average latency is reduced by 2.23 seconds, and the number of LLM calls is reduced by 0.82.It is worth noting that with the incorporation of skills, token consumption is reduced by 43% relative to the original setting. Experiment (1) exhibits a marginal decline in accuracy compared with Experiment (2), which we attribute to the fact that skill invocation occupies one round of LLM inference. Nevertheless, given the considerable cost advantage it affords, we regard this slight performance degradation as acceptable. Comparing experiments (2) and (3), we can see the role of the evaluation node in the reasoning process. Without the evaluation node to review past reasoning, the Acc@0.5 accuracy drops by 3.36%. Comparing experiments (3) and (4), we can see the effectiveness of the plan-act strategy. Compared with the strategy that adopts maximum score matching, planning and reasoning via plan-act improves the Acc@0.5 metric by 5.85%, which demonstrates that the strategy of planning before acting plays an important role in scene understanding and reflection. The comparison between experiments (1) and (5) aims to investigate the effectiveness of the uncertainty-aware scene graph. Through comparison, it can be found that the method that does not use the scene graph

but directly uses the scene memory at timestamp t_0 drops by 18.79% on the Acc@0.5 metric, which proves the effectiveness of the uncertainty-aware scene graph in understanding changing scenes.

Table 4: Study on the number of reasoning loops.

Loop Number	Acc@0.25 (%)	Acc@0.5 (%)
8	34.92	34.48
6	**34.98**	**34.51**
4	34.86	33.49
2	31.07	29.74
1	29.54	29.01

Table 5: Experimental results using different large language models.

Model	Acc@0.25 (%)	Acc@0.5 (%)
DeepSeek	**34.98**	**34.51**
GPT-5.2	34.12	33.83
Gemini-2.5-Pro	34.09	32.65
GLM-4.7	33.47	32.96

Table 4 presents the impact of different loop numbers on the final results, where one plan-act-evaluation inference cycle is defined as one loop. As observed, the grounding accuracy of the model gradually increases with the number of loops, and this growth plateaus when the number of loops reaches 6. Table 5 presents the experimental results using different LLMs. Although varying capabilities across different LLMs lead to distinct performance levels, the overall performance gap is not substantial. This proves that the localization capability of our method is not dependent on any single category of LLM.

Table 6: Results at Different Timestamps.

Timestamp	t_0	t_1	t_2	t_3	t_4	t_5	t_6	t_7
Acc@0.25 (%)	37.64	32.20	35.64	35.01	31.86	32.19	34.64	37.11
Acc@0.5 (%)	37.47	31.66	35.46	35.01	31.67	32.01	33.34	35.81

Table 6 demonstrates the grounding results under different timestamps. It can be observed that the grounding accuracy peaks at timestamp t_0, while subsequent timestamps exhibit lower accuracy than t_0. This is predictable, as the scene graph information at later timestamps might not be updated in a timely manner, thereby causing a decline in grounding accuracy. Notably, the accuracy does not linearly decrease with the increase of time, indicating that the erroneous information within the scene graph in our method does not gradually accumulate over time, which also demonstrates the performance stability of our model across different timestamps.

4.3 Qualitative Results

As shown in Figure 3, the upper portion illustrates the successful grounding results ($IoU > 0.5$), where green and blue boxes denote successfully predicted bounding boxes and GT object boxes, respectively. Only one green or blue bounding box can be observed in the visualization, since the predicted bounding boxes heavily overlap with the ground-truth boxes and cause occlusion. This phenomenon indirectly demonstrates the high accuracy of our object detection network: for objects correctly identified by the model, the deviation between predicted boxes and ground-truth boxes is minimal. We attribute this favorable performance to the programmatically generated scenes, where objects are arranged in an orderly manner. When FCAF3D performs convolution over voxels, each object can be fully partitioned into distinct voxels, which mitigates feature loss. This part reflects the model's capability to accurately comprehend referring expressions in the query and the spatial relations among objects.

Conversely, Figure 4 presents common failure cases of localization along with their underlying error causes, where red boxes denote incorrect predictions and blue boxes denote ground-truth boxes. Specifically, (a) shows the incorrect grounding result arising from the interference of similar objects in the scene. When multiple objects of the same category are placed close to one another in the scene, they may interfere with model prediction and lead to erroneous judgments by the agent. (b) displays the case where substantial misalignment between FCAF3D-predicted bounding boxes and ground-truth boxes degrades grounding performance. The bounding boxes output by FCAF3D may



Figure 3: Visualization images of grounding results with accuracy exceeding Acc@0.5 demonstrate the superiority of our method in the spatial understanding of embodied agents.

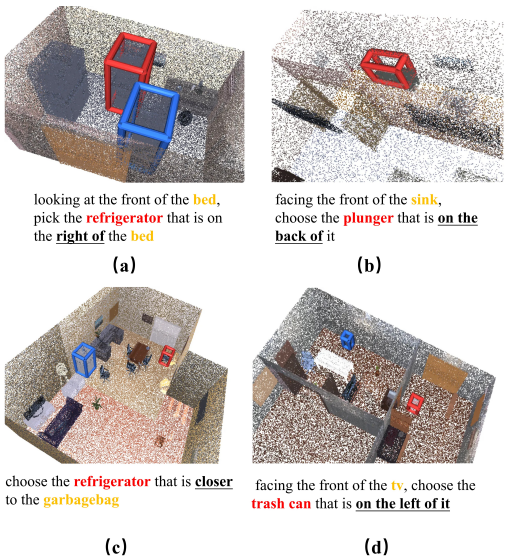

Figure 4: This figure illustrates four common causes of grounding failure, accompanied by corresponding visualization examples.

shift, resulting in inaccurate object prediction. Accordingly, the box selected by the agent according to the query will also be affected, and flawed bounding box predictions inevitably impair grounding results. (c) depicts the scenario where the scene graph fails to be updated promptly, causing the agent to rely on outdated information from earlier timestamps. In such cases, the bounding box corresponding to the target ground-truth object may remain unupdated. Meanwhile, the agent fails to notice the high uncertainty score of this object and performs reasoning based on stale scene graph data, which deteriorates localization performance. (d) illustrates that the agent suffers from LLM hallucinations when extracting spatial relations from the query. It incorrectly identifies the "left" relation as "right", thereby triggering localization failure. This error originates from inherent hallucination issues of large language models, which yield mistakes during structured parsing of textual queries.

5 Conclusion

In this paper, we introduced 3D Long-Term Changing Grounding, a novel task that extends 3D visual grounding from static environments to dynamic scenes. We constructed the 3D Long-Term Changing Dataset with ProcTHOR and Sr3D templates, capturing sequential object edits over continuous timestamps. To track evolving scenes with minimal exploration overhead, we proposed a detect-then-ground framework with a dynamic uncertainty-aware scene graph and an uncertainty-driven re-scanning strategy. We further developed an agent-based plan-act-evaluate reasoning module for structured LLM-based spatial localization, along with a skill-refinement mechanism to compress repetitive reasoning. Experiments show that our agent achieves stable accuracy across timestamps while maintaining controllable costs, and that specialized skills reduce latency with negligible accuracy loss, validating the effectiveness and robustness of our framework in dynamic real-world scenarios.

AI Use Statement

We used ChatGPT (OpenAI) to assist with literature retrieval and language polishing of the manuscript. All manuscript content was originally written by the authors. AI-generated sugges-

tions, including wording refinements and code hints, were thoroughly reviewed, verified, and edited by the authors, who take full responsibility for the integrity and accuracy of the contents of this paper.

REFERENCES

Panos Achlioptas, Ahmed Abdelreheem, Fei Xia, Mohamed Elhoseiny, and Leonidas Guibas. Referit3d: Neural listeners for fine-grained 3d object identification in real-world scenes. In *European conference on computer vision*, pp. 422–440. Springer, 2020.

Dave Zhenyu Chen, Angel X Chang, and Matthias Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. In *European conference on computer vision*, pp. 202–221. Springer, 2020.

Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 5828–5839, 2017.

Matt Deitke, Eli VanderBilt, Alvaro Herrasti, Luca Weihs, Kiana Ehsani, Jordi Salvador, Winson Han, Eric Kolve, Aniruddha Kembhavi, and Roozbeh Mottaghi. ProcTHOR: Large-scale embodied AI using procedural generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 35, pp. 5982–5994, 2022.

Miao Hu, Zhiwei Huang, Tai Wang, Jiangmiao Pang, Dahua Lin, Nanning Zheng, and Runsen Xu. Changinggrounding: 3d visual grounding in changing scenes. *arXiv preprint arXiv:2510.14965*, 2025.

Zhao Jin, Rong-Cheng Tu, Jingyi Liao, Wenhao Sun, Xiao Luo, Shunyu Liu, and Dacheng Tao. Spazer: Spatial-semantic progressive reasoning agent for zero-shot 3d visual grounding. *Advances in Neural Information Processing Systems*, 38:165549–165576, 2026.

Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4015–4026, 2023.

Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Matt Deitke, Kiana Ehsani, Daniel Gordon, Yuke Zhu, Ali Farhadi, and Hannaneh Hajishirzi. Ai2-thor: An interactive 3d environment for visual ai, 2017.

LangChain AI. Langgraph: Building language agents as graphs, 2024. URL `https://github.com/langchain-ai/langgraph`.

Rong Li, Shijie Li, Lingdong Kong, Xulei Yang, and Junwei Liang. Seeground: See and ground for zero-shot open-vocabulary 3d visual grounding. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pp. 3707–3717, 2025a.

Rong Li, Shijie Li, Lingdong Kong, Xulei Yang, and Junwei Liang. Zero-shot 3d visual grounding from vision-language models. *arXiv preprint arXiv:2505.22429*, 2025b.

Jiawen Lin, Shiran Bian, Yihang Zhu, Wenbin Tan, Yachao Zhang, Yuan Xie, and Yanyun Qu. Seqvlm: Proposal-guided multi-view sequences reasoning via vlm for zero-shot 3d visual grounding. In *Proceedings of the 33rd ACM International Conference on Multimedia*, pp. 3094–3103, 2025.

Tianhe Ren, Shilong Liu, Ailing Zeng, Jing Lin, Kunchang Li, He Cao, Jiayu Chen, Xinyu Huang, Yukang Chen, Feng Yan, et al. Grounded sam: Assembling open-world models for diverse visual tasks. *arXiv preprint arXiv:2401.14159*, 2024.

Danila Rukhovich, Anna Vorontsova, and Anton Konushin. Fcaf3d: Fully convolutional anchor-free 3d object detection. In *European Conference on Computer Vision*, pp. 477–493. Springer, 2022.

Johanna Wald, Armen Avetisyan, Nassir Navab, Federico Tombari, and Matthias Nießner. Rio: 3d object instance re-localization in changing indoor environments. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 7658–7667, 2019.

Runsen Xu, Zhiwei Huang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Vlm-grounder: A vlm agent for zero-shot 3d visual grounding. *arXiv preprint arXiv:2410.13860*, 2024.

Jianing Yang, Xuweiyi Chen, Shengyi Qian, Nikhil Madaan, Madhavan Iyengar, David F Fouhey, and Joyce Chai. Llm-grounder: Open-vocabulary 3d visual grounding with large language model as an agent. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7694–7701. IEEE, 2024.

Zhihao Yuan, Jinke Ren, Chun-Mei Feng, Hengshuang Zhao, Shuguang Cui, and Zhen Li. Visual programming for zero-shot open-vocabulary 3d visual grounding. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 20623–20633, 2024.

Yiming Zhang, ZeMing Gong, and Angel X Chang. Multi3drefer: Grounding text description to multiple 3d objects. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 15225–15236, 2023.

A Detailed Analysis of the Dataset

We statistically analyze query lengths as illustrated in Figure A.1.Query lengths exhibit a distinct bimodal distribution: short sentences cluster at approximately 8–12 words (peak at 10 words, 2,216 samples), while long sentences concentrate in the 16–22 word range (peak at 17 words, 4,104 samples). There are nearly no samples with 13–15 words in the intermediate interval. The average query length is around 15.1 words, with both the median and mode equal to 17. This pattern aligns with the template-based generation paradigm: allocentric descriptions (e.g., "looking at the front of the ...") tend to be longer, whereas concise relative-position statements are shorter. This indicates that the linguistic annotations encompass both brief referential phrases and comprehensive expressions with viewpoint constraints, rather than following a unimodal length distribution.

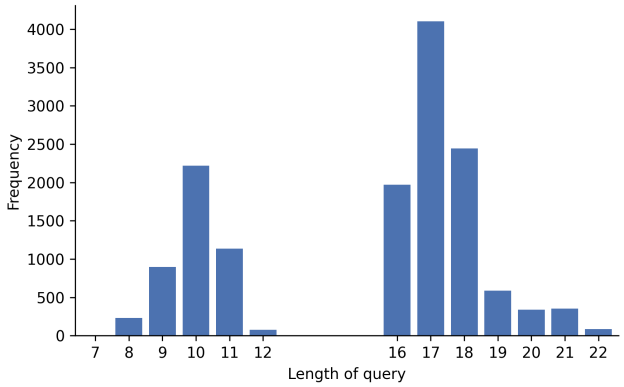

Figure A.1: Distribution of query lengths in the dataset.

In terms of referenced objects, Figure A.2 visualizes the word cloud of all referential objects contained in the queries, intuitively demonstrating that large-sized objects belonging to prevalent categories occupy a dominant proportion within the dataset. Figure A.3 illustrates the frequency statistics of target objects across all queries.There are 33 target object categories forming a classic long-tailed distribution. Large furniture and common appliances dominate, including *bed* (2,231), *chair* (1,726), *tv* (1,157), *dresser* (1,114), and *dining table* (1,091). The Top-5 / Top-10 target categories account for approximately 50.7% and 73.0% of all queries, respectively. The tail consists of low-frequency fine-grained objects such as *plunger*, *baseballbat*, *laundry hamper*, and *vacuum*.

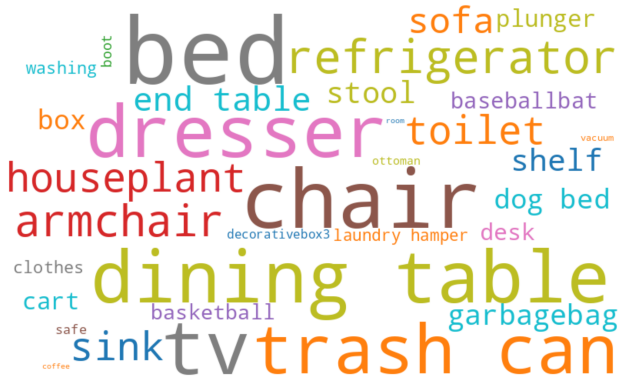

Figure A.2: Word cloud of referenced objects.

As shown in Figure A.4,the distribution of anchor categories is highly isomorphic to that of targets (also 33 categories), likewise dominated by *bed*, *tv*, *chair*, *dining table*, *dresser*, and similar items. This reflects the common referencing pattern where large furniture can serve as either targets or

spatial references. Multi-word categories in the word cloud (e.g., *dining table*, *trash can*) appear frequently as unified entities, consistent with the structured annotations.

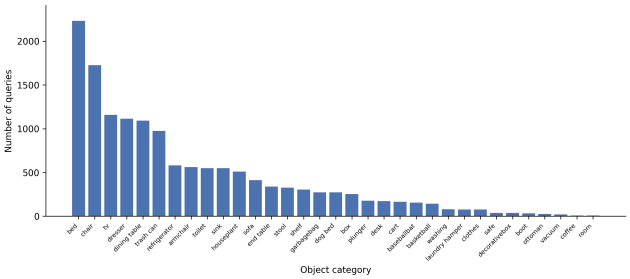

Figure A.3: Frequency distribution of target categories.

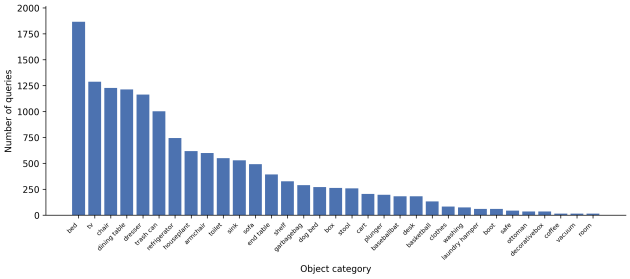

Figure A.4: Frequency distribution of anchor categories.

Regarding spatial relations, allocentric surface forms constitute the majority of spatial prepositions (approximately 89.7%). Front-back expressions including *in front of*, *behind / on the back of*, together with synonymous left-right formulations (*to/on the left/right of*, *on the left/right side of*) dominate the word cloud. Distance-related terms (*closer to*, *near*, *farthest from*, etc.) make up roughly 10.3%.

Overall, for spatial relations, allocentric expressions prevail, accompanied by horizontal distance descriptions. These statistics characterize the skewness and diversity of linguistic patterns and object distributions. They can serve as the primary evaluation benchmark for subsequent models on high-frequency categories, and highlight the necessity of investigating model robustness on long-tailed categories and diverse synonymous directional descriptions.

B Detailed Specifications of Skills Workflows

In this section, we present the formal operation specifications, triggering conditions, negative constraints, and output observations for the ten skills defined within our framework. Serving as structured prompt constraints, these detailed rules guide the agent to make rapid decisions during 3D spatial reasoning and reduce computational time overhead.

Box 1: Detailed Specifications and Implementation Rules for Ten Skill Workflows

1. **`wf_viewer_pose_relation`**
   - **Tool Pipeline:** `get_viewer_pose` → `compute_spatial_relation`
   - **Description:** Verify spatial relation from the anchor's front-facing viewpoint.
   - **Use when:** Preload top-1 candidate pair exists (target + anchor); query mentions viewpoint toward anchor (e.g., "facing", "looking at", "front of"); relation must be confirmed from that view.
   - **Do NOT use when:** No anchor candidate in preload, or relation was already computed in a prior observation.

- **After execution:** Observation includes `viewer_pose` (position, yaw) and computed relation for the top-1 pair.

2. **`wf_direct_final`**
   - **Tool Pipeline:** None
   - **Description:** Skip Loop-1 tool execution and proceed with reasoning over preloaded candidates only.
   - **Use when:** Preload top-1 already strongly matches query labels, spatial relation, and room constraints; query is straightforward with a single obvious target-anchor pair in top-3 candidates.
   - **Do NOT use when:** Top candidates have close confidence scores, belong to different rooms, or have ambiguous spatial relations; prior feedback suggests stale detections or missing objects.
   - **After execution:** No new tool observation; ReAct continues with preloaded context only.

3. **`wf_relation_only`**
   - **Tool Pipeline:** `compute_spatial_relation`
   - **Description:** Check spatial relation between top-1 target and anchor without changing viewpoint.
   - **Use when:** Top-1 pair labels and rooms look correct, but relation (left/right/front/back) needs confirmation; query does NOT require explicit "facing the anchor" verification.
   - **Do NOT use when:** Query emphasizes viewing direction from anchor front; target or anchor ID is uncertain.
   - **After execution:** Observation with relation, xz-distance, and `same_room` for top-1 pair.

4. **`wf_viewer_pose_only`**
   - **Tool Pipeline:** `get_viewer_pose`
   - **Description:** Obtain viewer pose relative to the anchor (position + yaw) only.
   - **Use when:** Query specifies "facing/looking at" the anchor and orientation is needed before further checks; next action will be decided in later loops.
   - **Do NOT use when:** Relation is already verified or query has no viewpoint requirement.
   - **After execution:** Observation with anchor position and `yaw_deg`.

5. **`wf_relation_relation`**
   - **Tool Pipeline:** `compute_spatial_relation` $\rightarrow$ `compute_spatial_relation`
   - **Description:** Verify spatial relation for two candidate pairs (typically top-1 and top-2).
   - **Use when:** Multiple preload candidates have similar scores or competing target-anchor pairs; need to compare which pair better satisfies the queried relation.
   - **Do NOT use when:** Only one plausible pair exists; labels/rooms already rule out the second candidate.
   - **After execution:** Two relation observations; compare which pair matches query relation.

6. **`wf_object_info_x6`**
   - **Tool Pipeline:** `get_object_info` $\times$ 6
   - **Description:** Inspect up to six object records (labels, rooms, bounding boxes) from top candidates.
   - **Use when:** Preload scores are ambiguous and label/room/bbox details are needed before choosing; suspected label noise (e.g., chair vs. armchair, garbagebag vs. trash can).
   - **Do NOT use when:** Top-1 is clear and relation-only check suffices; already inspected same objects in prior observation.
   - **After execution:** Multiple `get_object_info` observations for candidate objects.

7. **`wf_object_viewer_relation`**
   - **Tool Pipeline:** `get_object_info` $\rightarrow$ `get_viewer_pose` $\rightarrow$ `compute_spatial_relation`
   - **Description:** Inspect top target, then verify relation from anchor viewpoint.
   - **Use when:** Top-1 target label or room is uncertain; query includes viewpoint toward anchor AND relation must be verified.

- **Do NOT use when:** Top-1 labels are already trusted and only relation check is needed.
- **After execution:** Object info + viewer pose + relation for top-1 pair.

8. **`wf_object_object_relation`**
   - **Tool Pipeline:** `get_object_info` → `get_object_info` → `compute_spatial_relation`
   - **Description:** Inspect target and anchor objects, then compute their spatial relation.
   - **Use when:** Both target and anchor identity need confirmation before relation check; preload pair is plausible but labels/rooms need validation.
   - **Do NOT use when:** Only one side is uncertain (use `wf_object_info_only` or `wf_object_viewer_relation`).
   - **After execution:** Two object info blocks + relation observation.

9. **`wf_object_object_viewer_relation`**
   - **Tool Pipeline:** `get_object_info` × 2 → `get_viewer_pose` → `compute_spatial_relation`
   - **Description:** Full inspection of target and anchor, then viewpoint-based relation verification.
   - **Use when:** Both objects need inspection AND query specifies facing/looking at anchor; lighter skills failed or candidates remain tied after preload reasoning.
   - **Do NOT use when:** Simpler `wf_object_viewer_relation` or `wf_viewer_pose_relation` is sufficient.
   - **After execution:** Two object infos + viewer pose + relation observation.

10. **`wf_object_info_only`**
    - **Tool Pipeline:** `get_object_info`
    - **Description:** Look up a single object's label, room, and bounding box (typically top-1 target).
    - **Use when:** Need to confirm what object top-1 target actually is before relation or final answer; label mismatch suspected between query and graph label.
    - **Do NOT use when:** Relation or viewpoint verification is the main missing piece.
    - **After execution:** Single `get_object_info` observation.

C Prompt Design for LLM Inference

This section describes the prompts fed into the LLM. We adopt the Chat API, which requires inputs to be split into system and user roles. The system prompt contains fixed instructions, including agent role definition, execution rules, the full tool list, and standardized output formats. It remains nearly unchanged across multiple API calls for a single query and serves as global constraints imposed on the LLM. By contrast, the user prompt carries dynamic content that varies per sample, such as the raw query text, parsed fields, preloaded candidate objects, and historical observational results. Its content updates at every reasoning round after each tool invocation for the same query. In our pipeline, we concatenate the system prompt and user prompt as the complete input sent to the LLM.

Box 2: Prompt for 3D Temporal Grounding Agent

You are a 3D visual grounding agent on a **NOISY** predicted scene graph (`FCAF3D` detections).
Object labels and spatial relations in the graph are **IMPERFECT** — they will **NOT** match the query words exactly. Your job is to pick the best available target-anchor pair.

Scoring (used by candidate tools)

- `total_score = target_label_similarity + anchor_label_similarity + relation_vector_similarity`
- Higher `total_score` = better semantic fit. Relation and labels are fuzzy, not exact equality.

Temporal context

- `graph_timestep`: the maintained scene graph baseline (usually t_0).
- `gt_timestep`: the evaluation timestep — scan tools load `FCAF3D` detections from this timestep.

- At t_0 the graph is usually fresh; at $t > 0$ objects may have moved — scan if candidates look stale.

Workflow (at most 6 steps — be decisive)

1. Read the **PRELOADED** top-10 candidates below (already computed from the query).
2. Compare each candidate's graph labels, rooms, relation, and scores to the query text.
3. You may call `compute_spatial_relation` on 1–2 shortlisted pairs, or re-call `get_top5`/`get_top10` **ONLY** if you parsed target/anchor/relation differently.
4. If candidates are weak at $t > 0$, you may scan **ONCE**: `scan_room_from_gt(room_id)` or `scan_full_scene_from_gt()`, then `get_top5_matching_candidates` and decide.
5. Output **Final Answer** with your chosen `target_object_id` and `anchor_object_id`. The `target_id` **MUST** be one of the preloaded candidate `target_id`s unless you refreshed candidates with the same parsed labels/relation.

Rules

- **NEVER** expect graph labels to equal query words (e.g., query "trash can" → graph "otherfurniture").
- Do **NOT** call `list_objects_by_label` or `get_object_info` repeatedly — use the ranked candidates.
- Do **NOT** call `scan_*` on t_0 unless explicitly told the graph is stale.
- Prefer deciding within 3–4 steps; at most one manual scan per query.

ReAct format

```
Thought: ...
Action: tool_name
Action Input: {"key": "value"}
When done:
Final Answer: {"target_object_id": "...", "anchor_object_id":
"..."}
```

Available tools: Use `get_top5_matching_candidates` / `get_top10_matching_candidates` to rank pairs.

- `list_objects_by_label`: List all objects with a given category label in the current scene graph. *parameters:* `{"label": "object category, e.g. chair, bed"}`
- `get_object_info`: Return name, coordinates, bbox and room for one object id. *parameters:* `{"object_id": "unique object id"}`
- `get_object_room`: Return room id and room type for an object. *parameters:* `{"object_id": "unique object id"}`
- `get_viewer_pose`: Return viewer position and yaw when facing the front of an anchor object. Required for disambiguating left/right/front/back. *parameters:* `{"anchor_id": "anchor object id"}`
- `compute_spatial_relation`: Compute spatial relation from anchor to target. Supports left/right/front/back, above/below, closest/farthest, between. *parameters:* `{"target_id": "target object id", "anchor_id": "primary anchor object id", "second_anchor_id": "optional second anchor for between"}`
- `get_top5_matching_candidates`: Return the top-5 target-anchor pairs ranked by semantic score. Each candidate includes graph labels, rooms, computed relation, and score breakdown. Compare candidates to the query and choose one for Final Answer — labels need NOT match query words exactly. *parameters:* `{"target_label": "target category from query", "anchor_label": "anchor category from query", "relation": "expected spatial relation"}`
- `get_top10_matching_candidates`: Return the top-10 target-anchor pairs ranked by semantic score. Use when top-5 is ambiguous. Same fields as top-5. *parameters:* `{"target_label": "target category from query", "anchor_label": "anchor category from query", "relation": "expected spatial relation"}`
- `find_matching_targets`: Legacy helper: returns `best_match` plus ranked candidates (same scoring as `get_top10`). Prefer `get_top5`/`get_top10` for LLM decision making. *parameters:* `{"target_label": "target category from query", "anchor_label": "anchor category from query", "relation": "expected spatial relation"}`

- `diagnose_grounding_failure`: Diagnose why grounding failed: case1 (target missing), case2 (anchor missing), case3 (relation wrong). Returns hypothesis (added/moved/drift) and recommended scan action. *parameters:* {`"target_label": "target category", "anchor_label": "anchor category", "relation": "expected spatial relation"`}
- `scan_room_from_gt`: Update ONE room in the scene graph: replace that room's objects with predicted pre-box objects from `gt_timestep`. Other rooms unchanged. *parameters:* {`"room_id": "room id, e.g. room|4"`}
- `scan_full_scene_from_gt`: Replace the ENTIRE scene graph with predicted pre-box objects from `gt_timestep`. Use when object was added or moved across rooms. *parameters:* {}
- `run_recovery_pipeline`: One-shot: diagnose failure and automatically execute the recommended `scan_room` / `scan_full` recovery steps from `instructions.txt`. *parameters:* {`"target_label": "target category", "anchor_label": "anchor category", "relation": "expected spatial relation"`}

[Input Query Template]

```
Ground this query (graph_timestep={graph_timestep},
gt_timestep={gt_timestep}):
{description}
Parsed query fields: target={target_label},
anchor={anchor_label}, relation={relation}
PRELOADED top-10 candidates (use these for Final Answer
selection):
{compact_top10_json}
```

D FUNCTIONS AND IMPLEMENTATION DETAILS

Box 3: Tool Definitions and Execution Specifications

- **`get_top_n_matching_candidates`**
  Enumerates target–anchor pairs within the scene graph, sorts them based on the embedding similarity of textual labels, and returns the top-n candidate pairs.

- **`text_to_embedding`**
  Converts a specific textual term or label into a dense text embedding vector.

- **`get_viewer_pose`**
  Retrieves the observer's pose relative to the designated anchor object, which is subsequently utilized to rotate the point cloud to eliminate descriptive ambiguity.

- **`get_object_info`**
  Gets the complete and detailed spatial and semantic information of a single object.

- **`compute_spatial_relation`**
  Matches the identified coordinates of the target and anchor objects against the spatial relation specified in the query. Specifically:
  - **Between relation:** Verifies whether the target object lies on the line segment connecting the two anchor objects in the X–Z plane.
  - **Allocentric relation** (*e.g.*, left, right, front, back): Transforms the target coordinates into the anchor's local coordinate system to compare `local_x` and `local_z`.
  - **Vertical relation** (*e.g.*, above, below): Calculates the IoU of the 2D projected bounding boxes of the target and anchor objects; if IoU > 0.2, it compares their height difference to determine the vertical relationship.
  - **Distance relation** (*e.g.*, closest, farthest): Evaluates the Euclidean distance between the anchor and all candidate objects projected onto the horizontal plane within the room.

- **`scan_room` / `scan_full_scene`**
  Scans a specific room or the entire scene to dynamically update the maintained scene graph.

- **`find_target_score`**
  Calculates a total_score for auxiliary judgment:

$$\text{total_score} = \text{target_score} + \text{anchor_score} + \text{relation_score}$$

  Here, target_score and anchor_score serve as the similarity scores of text embeddings between the labels in the scene graph and the target and anchor objects in the query, respectively. relation_score captures the cosine similarity between the relative position vector of the two candidate objects in the scene graph and a standard spatial relation vector representing ideal conditions, such as $[-1, 0, 0]$ for left and $[0, 1, 0]$ for above.

- **`diagnose_grounding_failure`**
  Diagnoses the root causes of visual grounding failures and provides scanning recommendations. The identified failure categories include:

  (1) The target object mentioned in the query does not exist in the scene graph;

  (2) The target object exists, but the anchor object is missing;

  (3) Both target and anchor objects exist, but their spatial relation is incorrect.