ChangingGrounding: 3D Visual Grounding in Changing Scenes

Anonymous Author(s)
Affiliation
Address
email

Abstract

Real-world robots localize objects from natural-language instructions while scenes around them keep changing. Yet most of the existing 3D visual grounding (3DVG) method still assumes a perfectly reconstructed, up-to-date point cloud, an assumption that forces costly re-scans and hinders deployment. We argue that 3DVG should be reframed as an active, memory-driven problem, and we introduce ChangingGrounding, the first benchmark that explicitly measures how well an agent can exploit past observations, explore only where needed, and still deliver precise 3D boxes in changing scenes. To set a strong reference point, we also propose Mem-ChangingGrounder, a zero-shot framework that marries cross-modal retrieval with lightweight multi-view fusion: it identifies the object type implied by the query, retrieves relevant memories to guide attention, then explores the target in the scene based on this attention, gracefully falls back when previous operations are invalid, performs multi-view scanning of the target, and projects the fused evidence from multi-view scans into 3D space. We adapt three baselines to evaluate all approaches on ChangingGrounding. Mem-ChangingGrounder achieves the highest localization accuracy while relatively reducing exploration cost compared to baselines. We hope this benchmark and method catalyze a shift toward practical, memory-centric 3DVG for real-world applications. The codes, datasets, and benchmarks are available at `https://github.com/hm123450/ChangingGroundingBenchmark`.

1 Introduction

3D Visual Grounding (3DVG) is a critical technology that enables precise localization of target objects in 3D scenes through natural language instructions, with broad applications in service robotics [14], computer-aided room design [40, 13], and human-machine interaction [4, 23]. Current methodologies and benchmarks [2, 6] predominantly operate under static scene assumptions, where pre-reconstructed full scene point clouds [36] and textual queries [38] are fed into end-to-end models to predict 3D bounding boxes [16, 42, 27, 39, 15].

However, as shown in fig. 1, these approaches face significant limitations when deployed in real-world robotic systems: practical environments are inherently dynamic (e.g., furniture rearrangement, object occlusion/replacement). Existing methods require up-to-date full scene point clouds as input—a premise often infeasible in evolving scenarios due to two key challenges: (1) robots have to explore the whole scene again to reconstruct complete point clouds; (2) reconstructing point clouds also incurs substantial computational overhead. In stark contrast, humans searching in changing environments quickly draw on memories of past scenes to pinpoint likely target areas and can complete object localization through only a few new observations. Inspired by this insight, we contend that a new memory-based paradigm for real-world 3D visual grounding is needed.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

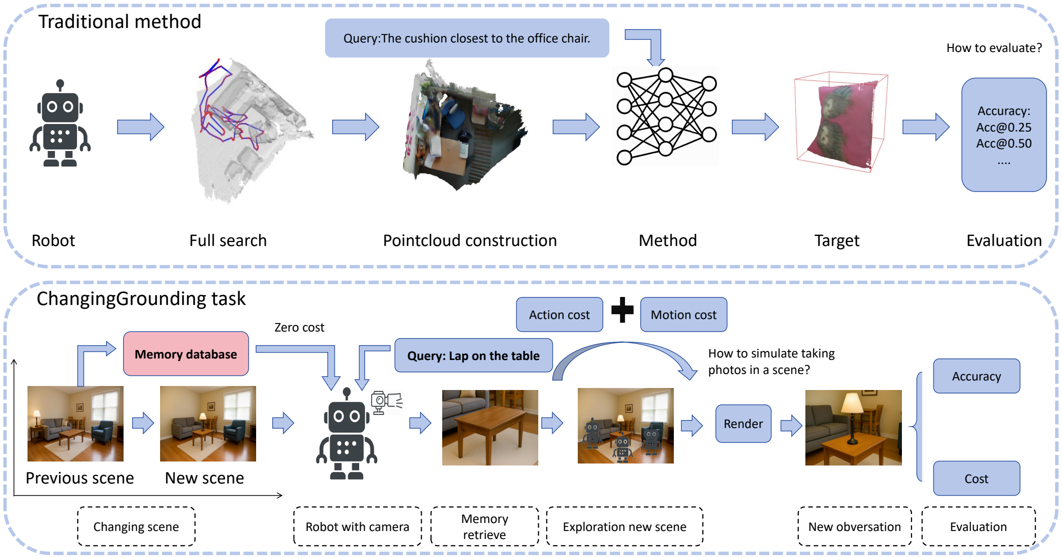


Figure 1: Comparison between traditional methods and ChangingGrounding task.

To the best of our knowledge, no existing work has explored 3D visual grounding in changing scenes by leveraging memory from past observations. In this paper, we formally define this task and introduce a novel benchmark called ChangingGrounding. The key motivation of the task and the benchmark is to measure how a 3D visual grounding system accurately and efficiently finds the target object by leveraging the memory of past observations and exploring the current scene.

We formally define the task as follows: given the memory of the previous scene, the unexplored current scene, and a query describing the target object in the current scene, the robot needs to predict the target's 3D bounding box in the current scene. We evaluate task performance using two key metrics: the accuracy of the predicted 3D bounding box and the cost for scene exploration. A better system achieves higher accuracy while keeping the lower cost of scene exploration. To support the task, we construct a ChangingGrounding dataset and benchmark, based on the 3RScan dataset [18] supported by a novel exploration and rendering pipeline to simulate how real-world robots perform 3D visual grounding.

In addition to our benchmark and dataset, we propose a novel framework called Mem-ChangingGrounder to address this new task. Our method is based on a previous zero-shot agent-based approach [43], due to the fact that current end-to-end approaches are not designed for memory access and scene agent exploration. Specifically, Mem-ChangingGrounder first classifies user queries, then retrieves relevant memories to guide its attention, and then explores the target images in the scene followed on this attention and the classification results, next ensures fallback localization if no valid target images are found, and finally performs multi-view scanning of the target and predicts 3D localization through multi-view projection.

Finally, we introduce three baseline methods and compare them with our proposed Mem-ChangingGrounder on the ChangingGrounding benchmark. The three baselines are: (i) Wandering Grounding: aimless exploration without memory, (ii) Central Rotation Grounding: simple exploration without memory, and (iii) Memory-Only Grounding: memory-only with no further exploration. These baseline methods and our method enable a more comprehensive evaluation of memory's contribution to the changing-grounding task under different collaborative modes. Experimental results show that Mem-ChangingGrounder boosts localization accuracy by 10%, while reducing exploration cost by 35%, achieving the best balance between accuracy and efficiency.

2 Related Work

3D Visual Grounding Benchmarks. 3D visual grounding locates target objects through natural language queries. Early work focused on object-level retrieval, matching objects to shape descriptions

[1, 35]. ScanRefer [6] and ReferIt3D [2] pioneer to establish scene-level 3D visual grounding benchmarks, constructing extensive natural language descriptions of 3D objects based on static point clouds from ScanNet [8]. A subtle distinction lies in their objectives: ScanRefer targets full grounding from queries to 3D bounding boxes, while ReferIt3D emphasizes correct object identification with pre-provided groundtruth candidate positions for selection. Furthermore, several datasets have made strides in aligning with real-world grounding situations. Multi3DRefer [47] introduced a benchmark for localizing multiple objects simultaneously, where a single natural language query may correspond to one or more target objects. ScanReason [48] introduced a benchmark using complex human instructions as queries instead of explicit object descriptions, highlighting human intention and reasoning ability. Despite their careful design to reflect real-world needs, these datasets neglect the temporal dimension, a critical aspect since real environments are changing.

3D Visual Grounding Methods. Previous 3D visual grounding methods can be broadly categorized into supervised end-to-end approaches and zero-shot methods. Supervised methods [15, 37, 42, 16, 27, 39] require training on annotated 3D scene datasets, using a 3D detection branch to extract candidate objects from the scene, followed by a language branch to encode the query text. The text features are then fused with object features to select the target object. While these methods perform well on benchmarks, their scalability is limited by the scarcity of annotated data. Recently, zero-shot methods leveraged Large Language Models (LLMs) [41, 9, 5, 29] and Vision-Language Models (VLMs) [30, 7, 24, 44] to understand scenes, addressing the data scarcity problem. Some methods [45, 46, 11] approach 3D visual grounding by reformulating the task into a text-based problem for LLM reasoning or using LLMs to write scripts for target grounding. VLM-Grounder [43] eliminates the reliance on complete scene point clouds by leveraging VLM to comprehend and ground objects within 2D images. SeeGround [21] uses a VLM to select viewpoints and renders the 3D scene into images for VLM input. However, none of the existing methods consider the situation of 3D visual grounding in changing scenes. As VLM-Grounder does not require reconstructed point clouds as input, the overall framework is more suitable for 3D visual grounding in changing scenes. We develop our method based on VLM-Grounder and also use this method for various baselines.

3D Perception in Changing Scenes. The academic dataset with scene changes can be traced back to the work of [12]. It was used to evaluate the performance of the 3D reconstruction algorithm in dynamic real-world scenes, however, it was relatively small in scale and lacked annotation information. InteriorNet [22] is a large-scale synthetic dataset that generates scenes with appearance and geometric variations by randomly simulating the movement of physical furniture and changes in lighting. 3RScan [18] pioneered the creation of a large-scale real-world indoor RGB-D dataset, encompassing scans of the same indoor environment at different time points, and introduced the task of 3D object instance relocalization, which involves relocating object instances within changing indoor scenes. Based on this dataset, several studies have begun to explore changing scene-understanding tasks, such as camera relocalization in changing indoor environments [19], changing detection [3], changing environment reconstruction [49], and changing prediction [26]. Besides, Hypo3D [28] conducts a 3D VQA benchmark to evaluate models' ability in changing scenes based on 3RScan. Notably, our work represents the first exploration of 3D visual grounding tasks in changing environments. The 3RScan dataset provides scene scans at different time steps, as well as the coordinate system transformations between scenes and the correspondences of objects. We construct our novel 3D visual grounding dataset based on these annotations.

In this section, we first formulate the ChangingGrounding task, then establish the evaluation metrics, and finally detail the dataset collection pipeline along with a statistical analysis.

2.1 Task Formulation

Consider a robot that observed a room yesterday and acquired its scene information. When revisiting the room today, where some changes like object rearrangements occurred, the robot is tasked to locate a target object specified by a user query. The naive solution requires full room exploration then applying standard 3D-VG methods, which is inefficient and not feasible for real-world deployment. Inspired by human ability to locate objects using memory, we propose enabling robots to similarly leverage previous memory for more efficient and accurate target grounding.

In this study, we formulate a novel visual grounding task in changing scenes. Given $\langle S_p, S_c, M_p, D_c \rangle$, the task is to predict 3D bounding box of target object. S_p refers to a previous physical scene that the robot visited. S_c denotes the currently unexplored scene when the robot revisits the same physical space. What has changed between S_c and S_p remains unknown. M_p encapsulates the robot's memory of the previous scene S_p, including RGB-D images, and their corresponding poses. D_c provides a textual description of a target object O in S_c. The task requires both efficient and precise grounding of the target object O in the scene S_c based on M_p and D_c. Therefore, we will evaluate this task by two key metrics: accuracy and exploration cost. Specifically, exploration cost comprises action cost C_a and motion cost C_m. (Further details are in section 2.2) In addition, for simplicity in research purposes, several task assumptions are established as follows.

Zero-cost memory access. The memory information M_p for the previous scene S_p is stored in the robot's database and can be accessed at any time without incurring additional cost.

Standardized scene coordinate system. Each 3D scene has been aligned to a standardized coordinate system T_s. For different temporal scene states of the same physical space, their standardized coordinate systems are aligned to one global coordinate.

Robot's Initial Pose. First, we decide to adopt the OpenCV right-handed camera coordinate convention and apply it to all poses. Therefore, for convenience, we simply assume that in each scene, the robot is initially positioned at the origin of T_s and its initial orientation is obtained by transforming T_s so that the axes satisfy the OpenCV convention

Exploration. For the new scene, S_c, the robot needs to explore to obtain relevant information about the scene. Therefore, the acquisition of information about S_c will involve certain costs. The specific definition of these costs will be detailed later in 2.2.

New observations. We assume the robot is equipped with an RGB-D camera and it can move to achieve new positions and orientations (new poses). At the new pose, the robot can obtain a new observation (an RGB-D image). To fulfill this assumption, we developed a rendering module. The rendering module takes the mesh file of a scene and the desired new pose as inputs and outputs the RGB-D image observed from the new pose within the scene.

$$(\mathbf{I}, \mathbf{D}) = \text{Rendering}(\text{Mesh}, Pose) \tag{1}$$

2.2 Evaluation metrics

The evaluation of the task is centered on two critical metrics: localization accuracy and cost. For localization accuracy, we follow the evaluation methodology employed in classic 3D-VG tasks, which is assessed by the ratio of samples for which the Intersection over Union (IoU) between the predicted 3D bounding box and the ground-truth bounding box exceeds a predefined threshold (e.g. Acc@0.25). As for the exploration cost, we consider action cost C_a and motion cost C_m as defined below.

C_a measures the number of actions required for the robot to move from its initial pose until successful target object localization. Each time the robot makes a decision and executes an action to reach a new pose to capture a new observation in the new scene, C_a increases by one.

However, considering only the action cost is sometimes insufficient to fully evaluate efficiency. In some cases, the action cost may be low, but the robot still performs a large amount of physical movement during execution (for example, a single action might involve moving forward by 20 meters), which can also lead to inefficiency. Therefore, inspired by the use of motion length as an evaluation metric in the navigation domain[17, 34], we also evaluate the robot's movement. Since a robot's motion consists of both translation and rotation, we take both into account in our evaluation, and we only consider translation within the horizontal plane.

Furthermore, to place translation and rotation on a common scale, we express the overall motion cost C_m in units of time by dividing each component by a nominal speed. Consistent with typical humanoid platforms, we assume a translational speed of $v = 0.5$ m/s and a rotational speed of $\omega = 1$ rad/s.

Formally, given the sequence of camera poses $\{(p_1, R_1), \ldots, (p_n, R_n)\}$ achieved until the target object is localized (n is therefore the total number of actions, i.e., C_a), the motion cost is:

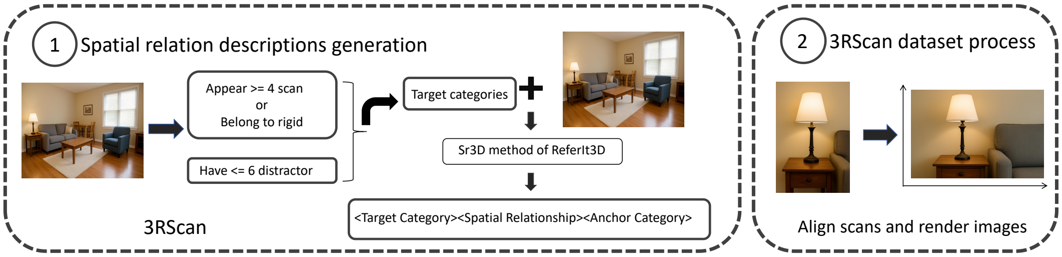

1
Spatial relation descriptions generation
Appear >= 4 scan
or
Belong to rigid
Have <= 6 distractor
Target categories
Sr3D method of ReferIt3D
<Target Category><Spatial Relationship><Anchor Category>
3RScan
2
3RScan dataset process
Align scans and render images

Figure 2: The Constructing pipeline for the ChangingGrounding dataset.

$$
\begin{aligned}
C_{\text{trans}} &= \frac{1}{v}\sum_{i=1}^{n-1}\|p_{i+1}-p_i\|, && v = 0.5\,\text{m/s},\\
C_{\text{rot}} &= \frac{1}{\omega}\sum_{i=1}^{n-1}\arccos\!\Big(\frac{\operatorname{Tr}\big(R_i^{\top}R_{i+1}\big)-1}{2}\Big), && \omega = 1\,\text{rad/s},\\
C_m &= C_{\text{trans}} + C_{\text{rot}}.
\end{aligned}
\tag{2}
$$

The rotation term uses the well-known trace formula $\theta = \arccos\big((\operatorname{Tr}(R^{\top}) - 1)/2\big)$, which gives the rotation angle θ of a rotation matrices (a direct consequence of Rodrigues' rotation theorem). By summing these angles and dividing by the nominal rotational speed ω, we obtain the rotation time.

2.3 Dataset and benchmark construction

To accommodate the proposed novel task, we constructed the ChangingGrounding dataset. The dataset includes the following components: (1)spatial relationship descriptions of the target objects, serving as user queries; (2)all the initial RGB-D images of each scene along with their corresponding camera poses, serving as memory information; (3)a mesh file for each scene, used to generate new observations. More specifically, we will construct ChangingGrounding dataset based on 3RScan dataset as it has 1,482 3D snapshots captured from 478 naturally changing indoor environments and provides transformation between different scans of the same scene which can help us to quickly align them to one global coordinate, dense instance-level semantic annotations, and correspondences of objects across scans which enables us to conveniently construct cases where the target object has been deliberately moved.

The detailed construction pipeline is illustrated in fig. 2, we first followed the method of ReferIt3D[2] to build a spatial relation descriptions set, then we performed processing on 3RScan original data to get global coordinates of scans and re-render the RGB-D images for improved usability.

Spatial relation descriptions set generation. We generate descriptions using a templated format: ⟨Target Category ⟩⟨Spatial Relationship ⟩⟨Anchor Category ⟩, such as "the chair farthest from the cabinet." The anchor category must differ from the target category. To obtain the target categories and the anchor categories, we begin by selecting 209 fine-grained object categories from the 3RScan dataset, defined as the union of categories that appear in at least four scenes and those labeled as rigid-move in the 3RScan metafile, which targets to maximise coverage of objects that change. A target is considered valid as in a scene if (a) it belongs to these 209 categories, and (b) the scene contains no more than six distractor objects of that class. Anchor object categories include these 209 classes plus 24 additional categories just like ReferIt3D. While ReferIt3D defines five spatial relationships including Horizontal Proximity, Vertical Proximity, Between, Allocentric, and Support, the Changing Grounding dataset excludes Allocentric relations due to the lack of front-orientation annotations in 3RScan, making it difficult to determine the correct front direction for objects with intrinsic orientation such as the back of an armchair.

3RScan dataset processing. We first align scans at different time instances for the same scene to a global coordinate system. Specifically, we select the initial scan as the reference scan and then construct its standardized coordinate system based on the floor for each scene, setting the origin at 0.25 meters above the center of the floor, and aligning the z-axis with the floor normal and the x-axis with the principal direction in the floor. After that, the transformation matrices between scans are applied to the standard coordinate system of the reference scan, allowing us to compute the standard coordinate systems of the remaining scans. After completing the alignment step, we focus on reprocessing the RGB-D images. Since the camera intrinsic, extrinsic parameters and image resolution in 3RScan are not commonly used, we establish a standardized camera model. In detailed, we adopt the released camera model parameters of ScanNet[8], (The resolution is 1296×968 and camera intrinsics is $(fx, fy, cx, cy) = (1169.6, 1167.1, 646.3, 489.9)$). Then, we use rendering module and re-renders all RGB-D images following standardized camera model in 3RScan dataset.

chair
shelf
pillow
box
lamp
curtain
window
door
table
kitchen cabinet
cabinet
plant
clothes
picture
bag
blanket
light
trash can
sink
monitor
doorframe
nightstand
stool
armchair
desk
clutter
counter
kitchen
basket
towel
heater
suitcase
wardrobe
cushion
ottoman
radiator
mirror
kitchen towel
windowsill
side table
bucket
stuffed animal
speaker
bath cabinet
commode
kettle
telephone
vase
rack
couch
coffee table
kitchen appliance
table lamp
showcase
bench
cupboard
pillar
frame
sofa
tv
stand
shoes
boxes
backpack

Figure 3: Word cloud of spatial relation descriptions set.

Statistics. The Changing Grounding dataset contains 266,916 referential language descriptions that can uniquely locate target objects through their spatial relationships with surrounding objects. As shown in fig. 3, The word cloud shows a gradual distribution of lexical prominence through varying font sizes, with larger fonts indicating higher-frequency terms. It captures both widely referenced pieces of indoor furniture and a rich assortment of lower-frequency household items. Also, the raw 3RScan dataset contains 528 fine-grained object categories. Many labels overlap or are seldom used, so a 528-class taxonomy is unnecessary in practice. To make the dataset more tractable for downstream use, we merge these labels into 225 broader, semantically coherent categories with the assistance of ChatGPT-o1 [33].

3 Mem-ChangingGrounder (MCG)

In this section, we introduce Mem-ChangingGrounder (MCG), a framework designed for 3D visual grounding in changing scenes. MCG processes the user query D_c in the current scene S_c, to predict a 3D bounding box of the target object O, with the help of the memory M_p of the previous scene, which are represented as RGB-D image sequences $\{I_p\}$ and their corresponding camera pose sequences $\{p_p\}$. As shown in Figure 4, the basic workflow of MCG is first to classify the user query to select the appropriate algorithm path for memory retrieval and grounding. If this stage fails to produce a reliable target, the fallback module is used. Finally, MCG will fuse the target's multi-view information to achieve accurate grounding. MCG is built upon VLM-Grounder[43], we will begin by introducing this framework (Section 3.1) and then present the details of MCG's four key modules.

3.1 Preliminary of VLM-Grounder

VLM-Grounder is a zero-shot 3D visual grounding method that achieves 3D localization of target objects using only 2D images and natural language descriptions. The main pipeline is as follows: all images containing the target category are detected and selected from the image sequence I_s scanning the whole scene to form I_s^{det}. Then, a VLM analyzes the user query and the stitched I_s^{det} to locate the image containing the target object. Next, an open-vocabulary object detector generates the target object proposals within the image, and the VLM selects the correct object. Finally, a multi-view ensemble projection module integrates information from multiple viewpoints to accurately estimate the 3D bounding box of the target object.

3.2 Overview and motivation of MCG key modules

Query classification. Because we find that different types of queries determine whether the system can rely on memory for direct grounding, which in turn affects the focus of memory retrieval, we introduce this module to classify user queries accordingly. The module sorts user queries into two groups: verifiable queries and non-verifiable queries.

Memory retrieval and grounding. This module is designed to combine memory and exploration to obtain an initial estimate of the target grounding result. It first either matches the anchor and target directly from memory or first pins down the anchor from memory depending on the classification results from prior module. After that, it make the robot explore further in current scene to pin

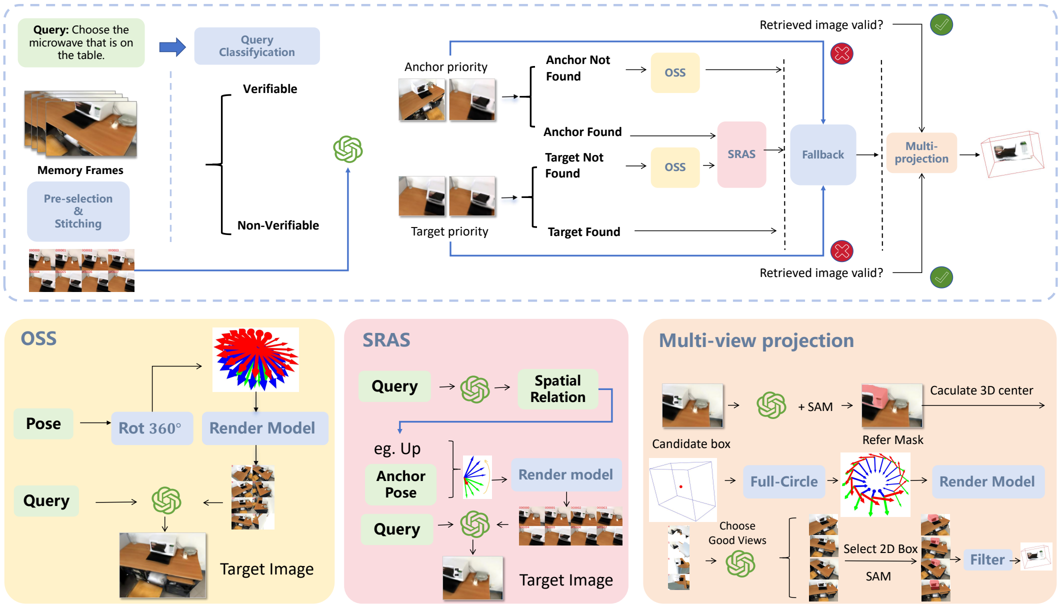


Figure 4: Workflow diagram of the Mem-ChangingGrounder(MCG). The upper part of the figure illustrates the overall pipeline of MCG. MCG first classifies user queries, then retrieves target object images via different memory retrieval and grounding algorithms based on the classification results of user queries, with help of OSS and SARS components, then ensures fallback localization if no valid target images are found, and finally predicts 3D bounding box of target through multi-view projection. The lower part of the figure provides details about Omnidirectional scene scanner(OSS), Spatial Relation Aware Scanner(SRAS), and multi-view projection in sequence. The OSS module receives a pose and a query, then predicts the target that matches the query through fixed-point 360-degree exploration. The SRAS module receives a anchor image pose and a query, then predicts the target image that matches the query through exploration starting from the anchor image pose guided by the spacial relationship of anchor and target indicated in the query. The multi-view projection module captures multiple-view images by rotating around the target object, then integrates and filters multi-view information to obtain a refined grounding result.

down the correct target in current scene. For convenience, we bundle particular exploration and procession into two reusable components and use them to determine the final target images: The **Omnidirectional Scene Scanner (OSS)** performs a quick 360° sweep to spot anchor or target images, whereas the **Spatial Relation Aware Scanner (SRAS)** module leverages the spatial relation between the anchor and the target to pinpoint the target object. More detailed information will be found in Section 3.3

Fallback. If memory retrieval and grounding module fails, the module will falls back to one target-class image with respective pose and use OSS to conduct a new search for target.

Multi-view projection. We build this module to obtain refined grounding results by collecting multi-view information and removing outliers. It first adopts a target-centered circular scanning strategy that captures multi-view observations of target object and then performs outlier filtering on their projections to get final grounding result.

3.3 Details of MCG

Note that the OSS and SRAS components may be reused across different modules of MCG, we first provide a detailed description of them then present the implementation details of MCG's four key modules.

Omnidirectional scene scanner. The OSS module receives a pose and a query, then predicts the target that matches the query through fixed-point 360-degree exploration. As shown in the first image bottom of fig. 4, given the initial pose p and user queries, OSS module will generate 20 poses by rotating p about its y-axis through $18^\circ \times i$ ($i = 0, 1, \ldots, 19$) followed by a downward rotation of 20° about x-axis. Then OSS module captures images at each pose, annotates sequential IDs, and dynamically stitches them. Finally, OSS will input stitched result to VLM to predict the correct image based on user queries.

$$\mathbf{p}_i = \mathbf{p} \cdot \mathbf{R}_y(18^\circ \times i) \cdot \mathbf{R}_x(-20^\circ), \quad i = 0, 1, \ldots, 19 \tag{3}$$

Spatial relation-aware scanner. The SRAS module receives a anchor image pose and a query, then predicts the target image that matches the query through exploration starting from the anchor image pose guided by the spacial relationship of anchor and target indicated in the query. As shown in the second image bottom of fig. 4, given the anchor image pose p_a and the user query D_c, VLM first analyzes the positional relationship between the target object o_t and the anchor object o_a based on the D_c. Leveraging this positional relationship, we adjust p_a to generate a series of new poses. Images captured under these new poses are annotated with IDs and dynamically stitched together. Finally, the stitched images and D_c are input into the VLM to predict the target image. For the support&vertical positional relationship, adjust p_a based on the relative position of o_t to o_a. If o_t is below o_a, first normalize p_a so that its z-axis aligns downward with gravity. Then rotate around its x-axis by 20 degrees each time to obtain a series of new poses. For the horizontal&between positional relationship, normalize p_a first, and then use a method consistent with the OSS module to process p_a to acquire a set of new poses.

Query classification. Because different types of queries determine whether the system can rely on memory for direct grounding when the scene is static, which determines whether memory retrieval should focus more on the target or the anchor, we need to classify queries first. As presented in fig. 4, user queries can be categorized into two types: verifiable queries and non-verifiable queries. A query is verifiable if, when we find matching anchor and target objects based on user query in S_p and these objects remain static in S_c, we can guarantee the target still satisfies the user query in S_c(e.g., "the vase on the table"). In contrast, non-verifiable queries are those where we can't guarantee the target objects matching user query in S_p will still satisfy same user query in S_c even if anchor and target objects don't move (e.g., "the chair farthest from the table" the target found in S_p could be wrong if a farther chair is added in S_c).

Memory retrieval and grounding. Two types of queries will have different methods for utilizing memory and grounding, we design this module to give memory-enhanced grounding solutions for both types of queries. After classify the query type, we can see in fig. 4, this module locates query-compliant target image and object in S_c by integrating memory M_p, user queries D_c, and exploration of S_c. Specifically, first following the VLM-Grounder approach, a 2D open-vocabulary detector filters M_p to generate a preprocessed image sequence $\{I^p\}_{seq}$ containing anchor class or target class, which are then dynamically stitched with ID annotations. For **non-verifiable queries**, the system prioritizes anchor object localization: VLM predicts anchor-containing image I_a^p and target-class image I_t^p for fallback purposes from $\{I^p\}_{seq}$, then compares I_a^p with its current scene counterpart I_a^c at the same pose p_a. If the anchor in images remains unchanged, module inputs I_a^p and its pose p_a into the Spatial relation-aware scanner (SRAS) for target retrieval; if anchor movement is detected, it initiates Omnidirectional scene scanner (OSS) at the center of S_c to directly locate the target. For **verifiable queries**, the module attempts direct target localization: VLM predicts anchor images I_a^p, query-matching target images $I_{target-match}^p$, and generic target images I_t^p from $\{I^p\}_{seq}$, then verifies the states of corresponding images I_a^c and $I_{target-match}^c$ in S_c: if both remain unchanged, it directly outputs $I_{target-match}^c$; if only the target changes but the anchor remains static, it invokes SRAS for re-localization; if the anchor moves, it uses OSS for the anchor position, and upon successful location, uses SRAS to track the target.

Fallback. If no valid target image is retrieved during memory retrieval and grounding module, the system will utilize the fallback target-class image I_t^p(as mentioned in the previous paragraph). Starting from the respective pose of this image, OSS will be activated to perform a 360° search for images containing the target object category.

Multi-scan projection. The projection point cloud from a single image may be incomplete due to the limited perspective, which can lead to insufficient localization accuracy. To address this, the module aims to capture multiple-view images by rotating around the target object with the initial target center as the focus. The specific process is as follows. Based on Memory-guided visual retrieval or Fallback target assurance, target images are acquired and input to the VLM to predict the 2D bounding box of the target object. The bounding box is subsequently fed into the SAM to generate masks, which are then projected using camera parameters and depth information to construct reference point clouds (mask and point clouds processing follows VLM-Grounder protocols). As presented in the last image bottom of fig. 4, we calculate the 3D bounding box center c and diagonal length l_{box} of the reference point cloud. Using the 3D bounding box center c and diagonal length l_{box} of the reference point cloud, an observation sphere is defined with center c and radius $center = c, r = \max(l_{box}/2, 1.5/m)$. Sixteen observation poses $\{T_i\}_{i=1}^{16}$ are uniformly deployed on a 30°-tilted equatorial plane, where each pose $T_i = (t_i, R_i)$ satisfies $\|t_i - c\| = r$ with the Z-axis of R_i pointing toward the sphere center and the Y-axis aligned to the gravity-vertical component. Then, images are captured at those poses through render model, dynamically stitched, and filtered by VLM to select 4 optimal frames. Next, 2D Open-vocabulary detection and SAM-based projection are performed on each frame to extract correct 2d sam whose corresponding point cloud's center has minimal Euclidean distance to c in candidates of one frame. We project all valid SAM results into 3D point clouds and sort them by bounding box size. We then filter outliers by removing any point cloud whose size exceeds 2.2 times that of the next smaller one - this eliminates cases where SAM incorrectly included background. Finally, we merge the remaining point clouds with the reference point cloud to obtain the final result.

4 Experimental Results

4.1 Experimental Settings

Dataset. The ChangingGrounding dataset is relatively large, so evaluating the entire set is unrealistic. Following the practice of VLM-Grounder[43] and LLM-Grounder[45], we randomly sampled 250 validation instances from the Changing Grounding dataset for evaluation, thereby keeping computational costs manageable. Each instance contains a query for the target object. Queries fall into two groups. "Unique" means that a single instance of the target class is in the scene, whereas "Multiple" means that additional same-class objects(distractors) are in the scene.

Baselines. Three baselines will be evaluated on 250 test samples. Note that when designing the baseline methods, we thoroughly considered two extreme scenarios: (i) relying exclusively on exploration with no memory, and (ii) relying exclusively on memory with no exploration. The three baselines are organized as follows: 1) Wandering Grounding: The original VLM-Grounder approach utilizing all captured images and corresponding poses of scene S_c provided by the 3RScan dataset for grounding; 2) Central Rotation Grounding: The VLM-Grounder utilizes images captured through similar methodology of PSS at the initial pose in scene S_c for grounding; 3)Memory-Only Grounding: VLM-Grounder exclusively uses images from the memory M_p in scene S_p for grounding.

Implementation Details. In our experiments, we use GPT-4.1-2025-04-14 [32] as VLM for baselines and MCG. We will conduct tests in both high-resolution and low-resolution image modes with the VLM. For VLM-Grounder variants, the VLM is configured with a temperature of 0.1 and a top-p value of 0.3 to balance randomness and creativity. We set the retry limit to $M = 3$, the maximum number of stitched images to $L = 6$, and the number of ensemble images to $N = 7$. For 2D open-vocabulary detectors, we used SAM-Huge [20, 10] and GroundingDINO [25]. The erosion kernel size is set to 7. For MCG, we adopt a similar configuration to the baseline, but without the retry limit (MCG has a different fallback mechanism).

Evaluation metrics. The accuracy evaluation metrics are $\mathrm{Acc}@0.25$ and $\mathrm{Acc}@0.5$, which represent the percentage of samples where the Intersection over Union (IoU) between the predicted bounding box and the ground-truth bounding box exceeds 0.25 or 0.50, respectively. The cost evaluation metrics include C_a and C_m (as defined in section 2.2).

4.2 Main Results

As shown in fig. 1, our Mem-Changing Grounder hereafter our method achieves the highest overall accuracy in both the low resolution and the high resolution settings of the VLM. The recorded figures are 29.2 percent and 36.8 percent respectively, and these values clearly surpass the three baseline approaches. That clear margin underlines the superiority and robustness of our solution for grounding performance across a spectrum of visual qualities. At the same time the method preserves modest numbers for the action cost C_a and the displacement cost C_m, which demonstrates a carefully engineered compromise between raw effectiveness and practical efficiency. This is because, using our MCG method, the robot consults its memory bank before moving and then performs brief, purposeful

Table 1: Accuracy and exploration cost of three baselines and Mem-ChangingGrounder(ours) on the ChangingGrounding Benchmark under both high-resolution and low-resolution settings. Different resolution setting is separated by a middle dividing line. The higher the accuracy and the lower the cost, the better the performance of the method. The highest method performance and the lowest cost are bolded. The cost in the table is measured in units of 1000 seconds.

			Overall		Unique		Multiple		Cost ↓			
Method	Model	Res	@0.25	@0.50	@0.25	@0.50	@0.25	@0.50	C_a	C_{trans}	C_{rot}	C_m
Wandering Grounding	GPT-4.1	low	24.80	10.80	**30.67**	10.67	16.00	11.00	44.48	9.88	8.61	18.49
Central Rotation Grounding	GPT-4.1	low	16.80	6.00	19.33	9.33	13.00	1.00	18.00	**0**	1.64	1.64
Memory-Only Grounding	GPT-4.1	low	20.80	10.00	22.67	10.67	18.00	9.00	**0.24**	0.56	**0.36**	**0.91**
Mem-Text Changing Grounder(ours)	GPT-4.1	low	**29.20**	**14.80**	30.00	**15.33**	**28.00**	**14.00**	8.53	5.84	3.98	9.82
Wandering Grounding	GPT-4.1	high	32.40	12.80	38.67	16.00	23.00	8.00	44.48	9.88	8.61	18.49
Central Rotation Grounding	GPT-4.1	high	17.20	6.80	18.00	8.00	16.00	5.00	18.00	**0**	1.64	1.64
Memory-Only Grounding	GPT-4.1	high	26.00	12.40	26.67	11.33	25.00	14.00	**0.24**	0.53	**0.38**	**0.91**
Mem-Text Changing Grounder(our)	GPT-4.1	high	**36.80**	**18.00**	**42.67**	**19.33**	**28.00**	**16.00**	8.47	5.97	3.92	9.88

Table 2: Accuracy and exploration cost of three baselines and Mem-ChangingGrounder(ours) on the ChangingGrounding Benchmark under both high-resolution and low-resolution settings. Different resolution setting is separated by a middle dividing line. The higher the accuracy and the lower the cost, the better the performance of the method. The highest method performance and the lowest cost are bolded. The cost in the table is measured in units of 1000 seconds.

			Overall		Unique		Multiple		Cost ↓			
Method	Model	Res	@0.25	@0.50	@0.25	@0.50	@0.25	@0.50	C_a	C_{trans}	C_{rot}	C_m
Wandering Grounding	GPT-4.1	low	24.80	10.80	**30.67**	10.67	16.00	11.00	44.48	9.88	8.61	18.49
Central Rotation Grounding	GPT-4.1	low	16.80	6.00	19.33	9.33	13.00	1.00	18.00	**0**	1.64	1.64
Memory-Only Grounding	GPT-4.1	low	20.80	10.00	22.67	10.67	18.00	9.00	**0.24**	0.56	**0.36**	**0.91**
Mem-Text Changing Grounder(ours)	GPT-4.1	low	**29.20**	**14.80**	30.00	**15.33**	**28.00**	**14.00**	8.53	5.84	3.98	9.82
Wandering Grounding	GPT-4.1	high	32.40	12.80	38.67	16.00	23.00	8.00	44.48	9.88	8.61	18.49
Central Rotation Grounding	GPT-4.1	high	17.20	6.80	18.00	8.00	16.00	5.00	18.00	**0**	1.64	1.64
Memory-Only Grounding	GPT-4.1	high	26.00	12.40	26.67	11.33	25.00	14.00	**0.24**	0.53	**0.38**	**0.91**
Mem-Text Changing Grounder(our)	GPT-4.1	high	**36.80**	**18.00**	**42.67**	**19.33**	**28.00**	**16.00**	8.47	5.97	3.92	9.88

translations or rotations to gather new observations, thereby avoiding lengthy exploratory loops that yield little new information.

In comparison, Wandering Grounding secures the second-highest accuracy at both resolution levels; however, its action cost is roughly five times larger, and its overall motion expense expands by a similarly substantial margin. The fundamental reason is that this method relies on wide-ranging roaming, compelling the robot to move repeatedly across the environment in extended back-and-forth sweeps. Although this exhaustive traversal allows the agent to accumulate a broader set of scene information and thereby enhance grounding accuracy, the very same coverage pattern obliges it to travel long distances and perform numerous actions, ultimately generating excessive overhead.

Central Rotation Grounding keeps the robot at the scene center and performs a single complete rotation, an action that virtually eliminates any need for translation and reduces the total number of discrete movements to a very small figure, making it an exceptionally low cost approach. Nevertheless, because the robot surveys the environment from this lone, inherently constrained vantage point, whenever an object happens to be obscured from view, situated at a noticeably elevated or depressed height, or when the overall spatial arrangement is particularly intricate, a substantial portion of crucial visual information never reaches the camera sensor. This resulting information gap ultimately pushes its aggregate grounding accuracy to the lowest position among all the methods that were examined.

Memory-Only Grounding likewise functions at an lowest cost because the entire inference process draws exclusively on previously stored panoramic memories; after estimating the most probable target location, the robot performs only one brief, final adjustment. In scenarios where those memories are fully comprehensive and the physical environment has remained unchanged, its grounding accuracy can be relatively high. However, if the surroundings have shifted in any way or if the original memory coverage contains gaps, the complete lack of a secondary verification step and the absence of any mechanism for active error correction trigger a pronounced drop in performance, ultimately placing this approach well behind our method.

When all evidence is taken together, our approach attains the highest grounding accuracy yet still keeps both action and motion costs firmly within manageable bounds, offering persuasive confirmation that memory-augmented strategies hold a decisive advantage in changing environments where stringent resource constraints demand a careful balance between efficiency and precision.

Similar to VLM-Grounder, our method exhibits a gap between Acc@0.25 and Acc@0.50, which stems from inheriting VLM-Grounder's core concept of projecting 2D images into 3D point clouds - a process where multiple potential noise sources exist, including the parameters, depth images, and both intrinsic and extrinsic camera matrices employed in this projection pipeline.

Table 3: Memory strategy

Strategy	@0.25	C_a	C_m
w/o. Memory	35.2	31.94	19.31
w. Memory	36.8	8.47	9.88

Table 4: Fall back

Fallback	@0.25	C_a	C_m
w/o. Fallback	36.4	8.21	9.66
w. Fallback	36.8	8.47	9.88

4.3 Ablation Studies

Memory Strategy Validation. To verify the effectiveness of our memory strategy, we conducted comparative experiments with a memory-free strategy where the system follows the original Wandering Grounding's pose sequence to explore anchor objects in scene S_c without utilizing any memory. As presented in the table3, the experimental results demonstrate that while both approaches achieve comparable accuracy levels, the memory-free solution incurs significantly higher costs, with its resource consumption dramatically exceeding that of our memory strategy.

Fallback. We conduct tests on our method after removing the fallback strategy. As presented in the table 4, the experimental results show that the accuracy and cost consumption with the fallback policy exhibit no significant difference compared to the method without it. However, from the perspective of system integrity, implementing the fallback policy ensures methodological completeness and enables coverage of edge cases in test samples.

Table 5: Multi-projection

Multi-view projection	@0.25	C_a	C_m
Baseline	22.4	4.81	3.06
+Multi-scan	28.0	8.52	9.85
+filter	36.8	8.47	9.88

Table 6: Different VLMs

VLMs	@0.25	C_a	C_m
GPT-4o	31.6	8.34	9.65
GPT-4.1	36.8	8.47	9.88

Multi-scan projection. To validate the effectiveness of key operations in the multi-view projection module, we conducted a two-step ablation study by sequentially adding operations to the baseline and observing the experimental results. As shown in the table 5, in the first step, we incorporated multi-view image acquisition through center rotation to the baseline. In the second step, we added the procedure for removing outliers in the multi-view ensemble point clouds. The results demonstrate that each operation contributes to significant accuracy improvements. Although the center rotation for multi-view acquisition incurs additional costs, it achieves a favorable balance between accuracy and cost by delivering substantial accuracy gains.

Different VLMs. We compared the performance of the Mem-ChangingGrounder using different VLMs on the test data, specifically testing GPT-4o [31] and GPT-4.1 [32]. As shown in the table 6, the difference in cost is not significant. However, GPT-4.1 achieves higher accuracy than GPT-4o [31]. This demonstrates that the capabilities of VLMs directly impact the performance of the Mem-ChangingGrounder.

5 Conclusion

In this work, we have reframed 3D visual grounding as an active, memory-driven problem and introduced ChangingGrounding—the first benchmark that couples changing scenes with explicit cost accounting—to foster research in this setting. Our Mem-ChangingGrounder demonstrates that leveraging memory and selective exploration can raise better localization accuracy while relatively cutting down exploration effort to baselines. We believe the dataset, task definition, and baseline suite released with this work will catalyze broader efforts toward deployable 3DVG, and we foresee future extensions that integrate continual memory updates, richer language understanding, and real-robot experiments in the wild. Details such as the full demo, failure analysis of the MCG modules, and a discussion of open problems will be provided in the supplementary material.

References

[1] Panos Achlioptas, Judy Fan, Robert X. D. Hawkins, Noah D. Goodman, and Leonidas J. Guibas. Shapeglot: Learning language for shape differentiation. *CoRR*, abs/1905.02925, 2019.

[2] 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 *ECCV*, 2020.

[3] Aikaterini Adam, Torsten Sattler, Konstantinos Karantzalos, and Tomas Pajdla. Objects can move: 3d change detection bynbsp;geometric transformation consistency. In *Computer Vision – ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXIII*, page 108–124, Berlin, Heidelberg, 2022. Springer-Verlag. ISBN 978-3-031-19826-7. doi: 10.1007/978-3-031-19827-4_7. URL `https://doi.org/10.1007/978-3-031-19827-4_7`.

[4] Charu C Aggarwal. A human-computer interactive method for projected clustering. *IEEE transactions on knowledge and data engineering*, 16(4):448–460, 2004.

[5] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. In *NeurIPS*, 2020.

[6] Dave Zhenyu Chen, Angel X Chang, and Matthias Nießner. Scanrefer: 3d object localization in rgb-d scans using natural language. In *ECCV*, 2020.

[7] Zhe Chen, Jiannan Wu, Wenhai Wang, Weijie Su, Guo Chen, Sen Xing, Muyan Zhong, Qinglong Zhang, Xizhou Zhu, Lewei Lu, Bin Li, Ping Luo, Tong Lu, Yu Qiao, and Jifeng Dai. Internvl: Scaling up vision foundation models and aligning for generic visual-linguistic tasks. In *CVPR*, 2024.

[8] Angela Dai, Angel X Chang, Manolis Savva, Maciej Halber, Thomas Funkhouser, and Matthias Nießner. Scannet: Richly-annotated 3d reconstructions of indoor scenes. In *CVPR*, 2017.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. 2021.

[11] Jiading Fang, Xiangshan Tan, Shengjie Lin, Igor Vasiljevic, Vitor Guizilini, Hongyuan Mei, Rares Ambrus, Gregory Shakhnarovich, and Matthew R Walter. Transcrib3d: 3d referring expression resolution through large language models, 2024.

[12] Marius Fehr, Fadri Furrer, Ivan Dryanovski, Jürgen Sturm, Igor Gilitschenski, Roland Siegwart, and Cesar Cadena. Tsdf-based change detection for consistent long-term dense reconstruction and dynamic object discovery. pages 5237–5244, 2017. doi: 10.1109/ICRA.2017.7989614.

[13] Yaroslav Ganin, Sergey Bartunov, Yujia Li, Ethan Keller, and Stefano Saliceti. Computer-aided design as language. *Advances in Neural Information Processing Systems*, 34:5885–5897, 2021.

[14] Juan Angel Gonzalez-Aguirre, Ricardo Osorio-Oliveros, Karen L Rodríguez-Hernández, Javier Lizárraga-Iturralde, Ruben Morales Menendez, Ricardo A Ramirez-Mendoza, Mauricio Adolfo Ramirez-Moreno, and Jorge de Jesus Lozoya-Santos. Service robots: Trends and technology. *Applied Sciences*, 11(22):10702, 2021.

[15] Wenxuan Guo, Xiuwei Xu, Ziwei Wang, Jianjiang Feng, Jie Zhou, and Jiwen Lu. Text-guided sparse voxel pruning for efficient 3d visual grounding. In *CVPR*, 2025.

[16] Ayush Jain, Nikolaos Gkanatsios, Ishita Mediratta, and Katerina Fragkiadaki. Bottom up top down detection transformers for language grounding in images and point clouds. In *ECCV*, 2022.

[17] Steeven Janny, Hervé Poirier, Leonid Antsfeld, Guillaume Bono, Gianluca Monaci, Boris Chidlovskii, Francesco Giuliari, Alessio Del Bue, and Christian Wolf. Reasoning in visual navigation of end-to-end trained agents: a dynamical systems approach. In *Proceedings of the Computer Vision and Pattern Recognition Conference*, pages 12111–12121, 2025.

[18] Nassir Navab Federico Tombari Matthias Niessner Johanna Wald, Armen Avetisyan. Rio: 3d object instance re-localization in changing indoor environments. 2019.

[19] Stuart Golodetz Tommaso Cavallari Federico Tombari Johanna Wald, Torsten Sattler. Beyond controlled environments: 3d camera re-localization in changing indoor scenes. In *European Conference on Computer Vision (ECCV)*, 2020.

[20] 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 *ICCV*, 2023.

[21] 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 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2025.

[22] Wenbin Li, Sajad Saeedi, John McCormac, Ronald Clark, Dimos Tzoumanikas, Qing Ye, Yuzhong Huang, Rui Tang, and Stefan Leutenegger. Interiornet: Mega-scale multi-sensor photo-realistic indoor scenes dataset. In *British Machine Vision Conference (BMVC)*, 2018.

[23] Yong-Lu Li, Xinpeng Liu, Han Lu, Shiyi Wang, Junqi Liu, Jiefeng Li, and Cewu Lu. Detailed 2d-3d joint representation for human-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10166–10175, 2020.

[24] Haotian Liu, Chunyuan Li, Qingyang Wu, and Yong Jae Lee. Visual instruction tuning. In *NeurIPS*, 2023.

[25] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *ECCV*, 2024.

[26] Samuel Looper, Javier Rodriguez-Puigvert, Roland Siegwart, Cesar Cadena, and Lukas Schmid. 3d vsg: Long-term semantic scene change prediction through 3d variable scene graphs. IEEE International Conference on Robotics and Automation (ICRA), 2023. doi: 10.1109/ICRA48891.2023.10161212.

[27] Junyu Luo, Jiahui Fu, Xianghao Kong, Chen Gao, Haibing Ren, Hao Shen, Huaxia Xia, and Si Liu. 3d-sps: Single-stage 3d visual grounding via referred point progressive selection. In *CVPR*, 2022.

[28] Ye Mao, Weixun Luo, Junpeng Jing, Anlan Qiu, and Krystian Mikolajczyk. Hypo3d: Exploring hypothetical reasoning in 3d. *ICML*, 2025.

[29] OpenAI. Gpt-4 technical report. *arXiv:2303.08774*, 2023.

[30] OpenAI. Gpt-4v. `https://openai.com/index/gpt-4v-system-card/`, 2023.

[31] OpenAI. Gpt-4o. `https://openai.com/index/hello-gpt-4o/`, 2024.

[32] OpenAI. Gpt-4.1. `https://openai.com/index/gpt-4-1/`, 2025.

[33] OpenAI. Openai-o1. `https://openai.com/o1/`, 2025.

[34] Nikhilanj Pelluri. Transformers for image-goal navigation. *arXiv preprint arXiv:2405.14128*, 2024.

[35] Mihir Prabhudesai, Hsiao-Yu Tung, Syed Ashar Javed, Maximilian Sieb, Adam W Harley, and Katerina Fragkiadaki. Embodied language grounding with implicit 3d visual feature representations. *CVPR*, 2020.

[36] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *NeurIPS*, volume 30. Curran Associates, Inc., 2017.

[37] Zhipeng Qian, Yiwei Ma, Zhekai Lin, Jiayi Ji, Xiawu Zheng, Xiaoshuai Sun, and Rongrong Ji. Multi-branch collaborative learning network for 3d visual grounding, 2024. URL `https://arxiv.org/abs/2407.05363`.

[38] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. In *ICML*, 2021.

[39] Xiangxi Shi, Zhonghua Wu, and Stefan Lee. Viewpoint-aware visual grounding in 3d scenes. In *CVPR*, pages 14056–14065, 2024. doi: 10.1109/CVPR52733.2024.01333.

[40] Michael A. Sipe and David Casasent. Feature space trajectory methods for active computer vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(12):1634–1643, 2003.

[41] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[42] Yanmin Wu, Xinhua Cheng, Renrui Zhang, Zesen Cheng, and Jian Zhang. Eda: Explicit text-decoupling and dense alignment for 3d visual grounding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023.

[43] Runsen Xu, Zhiwei Huang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Vlm-grounder: A vlm agent for zero-shot 3d visual grounding. In *CoRL*, 2024.

[44] Runsen Xu, Xiaolong Wang, Tai Wang, Yilun Chen, Jiangmiao Pang, and Dahua Lin. Pointllm: Empowering large language models to understand point clouds. In *ECCV*, 2024.

[45] 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. 2024.

[46] 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 *CVPR*, 2024.

[47] 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 (ICCV)*, pages 15225–15236, October 2023.

[48] Chenming Zhu, Tai Wang, Wenwei Zhang, Kai Chen, and Xihui Liu. Scanreason: Empowering 3d visual grounding withnbsp;reasoning capabilities. In *Computer Vision – ECCV 2024: 18th European Conference, Milan, Italy, September 29–October 4, 2024, Proceedings, Part VIII*, page 151–168, Berlin, Heidelberg, 2024. Springer-Verlag. ISBN 978-3-031-73241-6. doi: 10.1007/978-3-031-73242-3_9. URL `https://doi.org/10.1007/978-3-031-73242-3_9`.

[49] Liyuan Zhu, Shengyu Huang, and Iro Armeni Konrad Schindler. Living scenes: Multi-object relocalization and reconstruction in changing 3d environments. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024.

Appendix Overview and Organization

This appendix provides supplementary details to support and extend the main paper. The organization of the appendix is as follows:

1. **Benchmark Statement (Section A):** This section outlines the release status and usage policy of the our ChangingGrounding benchmark (CGB), and our Mem-ChangingGrounder (MCG) method are also provided to facilitate replication and validation of results.
2. **Spatial Relation-Aware Scanner (Section ??):** This section presents a detailed description of the SARS module. It explains how the system interprets spatial relationships in user queries and adjusts camera viewpoints accordingly to retrieve the most informative target views.
3. **Multi-Scan Projection (Section ??):** Inspired by VLM-Grounder, this section introduces the multi-scan projection approach, which combines multi-view point clouds. It elaborates on how reference point clouds are obtained and how noisy candidates are filtered out using distance constraints and language priors.
4. **VLM Prompts (Section ??):** We provide the full list of vision-language prompts used in MCG, covering all modules including memory retrieval, spatial relation parsing, multi-view comparison, and fallback strategies. These prompts form a modular, interpretable interface for multi-stage reasoning.
5. **Cost Calculation for Methods (Section ??):** This section details how exploration trajectories and motion costs are computed for each method. The evaluation aligns with the cost metrics defined in the main text, and a note explains that all costs are reported in units of 1,000 seconds (e.g., 9k = 9000s).
6. **More Results (Section ??):** Additional results are presented to assess the robustness of MCG, including a comparison between using rendered vs. real images in memory, and a set of failure cases analyzing the limitations of VLM, SARS, SAM, and the projection pipeline. A complete example is shown to illustrate how MCG grounds a target object in a dynamic scene.
7. **Open problems (Section ??):** We outline the current limitations of the CGB benchmark and the MCG method, including the lack of allocentric relations, the impact of rendering noise, and the dependency on external 2D models. Future improvements are discussed.
8. **Broader Impact (Section ??** The broader societal implications of our work are discussed, including potential applications in robotics and automation, ethical considerations, and the importance of open-source transparency for reproducibility and fairness.

A Benchmark statement

We publicly release the proposed CGB benchmark and its accompanying dataset on the Huggingface platform, making it freely accessible to the research community. The dataset will be regularly updated and maintained to ensure its accuracy and relevance. We hope this benchmark will encourage further research into 3D visual localization in dynamically changing environments.

All files within the CGB benchmark are strictly intended for non-commercial research purposes and must not be used in any context that could potentially cause harm to society.

To support reproducibility, we also provide benchmark testing examples for the proposed MCG method, along with detailed environment specifications and a complete execution pipeline to facilitate efficient replication and verification of experimental results. It is important to note that, at this stage, all available data in the CGB benchmark is used exclusively for testing purposes.

B Spatial relation-aware scanner

This section provides a more detailed description of the Spatial relation-aware scanner (SRAS). SRAS accepts a user query with an anchor-object image and its corresponding camera pose, and outputs images containing the target object while satisfying the specified query.SRAS operates in three main stages: (1) it analyzes the spatial relations specified in the user query; (2) it selects subsequent camera poses according to those relations; and (3) it feeds the newly captured images into the VLMs to identify the correct target image that matches the query.

B.1 Spatial relation parsing

The user query, along with the target object category and anchor object category, is provided as input to the Vision-Language Models (VLMs), allowing them to infer the spatial relationship of the target object relative to the anchor object described in the query. For example, given the instruction "Please select the vase on the table," the VLMs would respond with "above," since the target object (vase) is positioned above the anchor object (table).

B.2 Camera pose planning

Based on the inferred spatial relationship, the camera pose of anchor-object image will be adjusted to obtain a series of new camera poses. Here, we uniformly define the camera pose orientation such that the Z-axis points into the image, the X-axis points to the right, and the Y-axis points downward. **Up.** First, the camera pose corresponding to the image containing the anchor object is normalized by aligning its Y-axis with the gravity direction. The camera is then translated slightly backward along the Z-axis to ensure a wider field of view. Next, the pose is rotated around its local Y-axis (which controls the left-right viewing direction) at multiple angles. For each of these Y-axis rotations, additional rotations are applied around the local X-axis (which controls the up-down viewing direction) in an upward direction to generate a set of upward-looking viewpoints. These poses collectively form a diverse set of upward trajectories. **Down.** The "down" case follows a process highly similar to the "up" case, with the key difference being the direction of rotation around the local X-axis (which controls the up-down viewing direction). Instead of rotating upward, the camera is rotated downward to produce a set of overhead viewpoints. This generates a diverse set of downward-looking trajectories. **Horizontal and between.** For spatial relations such as “horizontal” and “between,” the system first slightly translates the camera pose along its local Z-axis to obtain a broader field of view. It then interpolates the position to move the camera slightly closer to the center of the room, further expanding the observable area. Next, the system performs a 360-degree rotation around the camera’s local Y-axis (which controls the left-right viewing direction) to generate a diverse set of viewpoints covering various horizontal angles. At each rotation step, the camera is also slightly tilted downward along its local X-axis to better capture lower portions of the scene.

B.3 Target image verification

After obtaining a series of newly generated poses, we feed them into a rendering model to simulate corresponding camera views and generate images. These images are then stitched together and passed into VLMs, which identify the image that best matches the user’s query.

C Multi-scan projection

Inspired by VLM-Grounder, the multi-scan projection approach also aggregates multi-view point clouds to obtain the final point cloud. The entire pipeline can be divided into three stages: (1) obtaining a reference point cloud, (2) performing surround-view scanning to collect multi-view point clouds, and (3) removing outliers from the aggregated point cloud set. In the main text, we have clearly described the overall pipeline of the multi-view projection. Here, we provide a more detailed explanation of certain specific steps in the process.

Reference point cloud. First, the image containing the target object is fed into GroundingDINO for object detection, and overly large 2D bounding boxes are filtered out (as GroundingDINO may occasionally return boxes covering the entire image). The centers of the remaining candidate boxes are then sequentially marked and passed into a vision-language model (VLM). The VLM, guided by the user query and additional contextual cues (e.g., "the object is located in the bottom-right corner"), identifies the most semantically relevant 2D bounding box corresponding to the target object. This selected bounding box, along with its center point—which likely lies on the object due to the VLM’s judgment—is then used as a positive point input to the SAM model to obtain a reference segmentation mask. Finally, the mask is projected into a 3D point cloud using the camera parameters and the depth image, with the same denoising strategy of VLM-Grounder during projection.

Removing outliers. In addition to the outlier removal strategy based on bounding box size sorting described in the main text, we also apply an initial denoising step during candidate box selection. Specifically, for a given view, we first extract all 2D candidate boxes and use them as prompts for SAM to generate segmentation masks. These masks are then projected into 3D point clouds. Among the resulting candidates, we select the one whose center is closest to that of the reference point cloud. However, due to the limitations of the 2D object detector and SAM, this nearest candidate may not always correspond to the true target object. To address this, we first input the reference image into a vision-language model (VLM) to assess whether the target object is particularly large or partially outside the camera view. If so, no additional filtering is applied. Otherwise, we enforce a spatial constraint requiring that the center of the selected candidate point cloud lies within 0.25 meters of the reference center; this helps prevent the inclusion of significant noise points unrelated to the target object.

D Open problems

We present CGB as the first benchmark for evaluating 3D visual grounding in dynamically changing scenes and introduce MCG as a strong reference method. Nevertheless, both still exhibit the following limitations.

D.1 Limitations of the CGB Benchmark

At present, our CGB dataset models only the relative positional changes between the target and its surroundings, without accounting for critical factors such as lighting variations, object appearance attributes (e.g., color, material, deformation), or dynamic scene interactions. Moreover, its repertoire of spatial relations lacks allocentric descriptions like “Object A is in front of Object B.” These omissions narrow the benchmark’s breadth and depth when assessing an agent’s cross-scene generalization and robustness. Future work can address these gaps by enriching multimodal annotations, introducing additional dimensions of variation, and incorporating allocentric relations, thereby expanding the dataset’s scale and diversity and enhancing CGB’s applicability and challenge in real-world dynamic environments.

D.2 Limitations of the MCG method

Limitations of VLM capability. MCG relies heavily on the underlying Vision–Language Model (VLM) to locate target objects in image sequences according to the analysis requirements. As demonstrated by the ablation studies above, the strength of the VLM has a decisive impact on MCG’s final grounding accuracy. If the VLM is insufficiently capable—or if the visual information in real-world scenes is unusually complex—MCG’s performance can deteriorate. Nevertheless, because VLM technology is advancing rapidly, we can replace the current module with more powerful models in the future to further enhance performance. **Noise from rendered images.** During the experiments, MCG consistently feeds rendered RGB-D images into the vision-language model (VLM) for inference, or uses them for SAM-based segmentation, projection, and related processes. However, the rendering process based on mesh files introduces various types of noise, including artifacts in the RGB images and inaccuracies in the depth maps. Moreover, there may be inherent differences in how VLMs process real versus rendered images. These factors can negatively affect the grounding accuracy. **Noise introduced by 2D models.** MCG depends on 2-D object detectors and segmentation networks to filter candidate images and perform the final projection. Although state-of-the-art models such as GroundingDINO and SAM are highly capable, they still exhibit missed detections, false positives, imprecise bounding boxes, and segmentation errors. These imperfections propagate through the pipeline and ultimately undermine the accuracy of the grounding results. **Future work** Despite these limitations, we believe that our work on MCG and the CGB benchmark provides a strong foundation for future research in the field of grounding task in changing scene. We hope that our contributions will inspire researchers to explore new methods and techniques to address the challenges posed by dynamic scenes. Specifically, we encourage the community to focus on the following open problems: (1) Improving VLM Robustness: Developing more robust Vision–Language Models that can handle complex real-world visual information and reduce the impact of noise; (2) Enhancing Multimodal Integration: Exploring ways to better integrate multimodal data (e.g., combining visual, linguistic, and spatial information) to improve grounding accuracy; (3)Expanding Benchmark Diversity: Contributing to the expansion of the CGB benchmark by adding more diverse scenarios, including variations in lighting, object appearance, and dynamic interactions; (4)Reducing Noise in Rendered Data: Investigating methods to minimize the noise introduced during the rendering process and to bridge the gap between real and rendered images; (5)Advancing 2D-to-3D Projection Techniques: Improving the accuracy and reliability of 2D object detection and segmentation models to enhance the overall grounding performance. We hope that our work will serve as a catalyst for further research in this exciting and challenging domain. By addressing these open problems, we can collectively push the boundaries of 3D visual grounding in changing environments and develop more effective and robust solutions.

E Broader impact

This study systematically introduces a new task for changing scene 3D visual grounding task, releasing the open benchmark CGB together with a strong reference method, MCG. The proposed technology could markedly improve logistics and service robots' perception–decision loop efficiency in complex, dynamic environments, thereby accelerating upgrades in smart manufacturing and supply-chain management. Nevertheless, rapid advances in automation may, in the short term, displace low-skill positions such as manual handling and sorting, prompting shifts in the employment structure. To balance technological progress with social inclusiveness, we urge governments and industry to launch joint reskilling programmes that equip affected workers with digital competencies. To lower the research threshold and enable independent fairness audits, we have fully open-sourced our code, data, and evaluation scripts under the MIT license; all data are drawn from public or simulated scenes and contain no personally identifiable information. We believe that open-source collaboration and responsible governance will allow dynamic 3D visual grounding technology to serve society fairly and sustainably.

F VLM prompts

For the baseline methods, we use the same prompts as those employed in VLM-Grounder. For the MCG method, we introduce several additional prompts, including those designed for the memory retrieval image module, prompts used to compare whether the target object has moved between images., prompts used in SRAS, and prompts applied in the multi-scan projection process. We will explain each of them in the following sections.

The **memory_retrieval_prompt_for_unverifiable_queries** selects the top 3 images that clearly capture the anchor object from a video sequence when no reliable grounding information is available. In contrast, the **memory_retrieval_prompt_for_verifiable_queries** performs a two-stage reasoning process: it first searches for images that satisfy the query constraints and falls back to identifying the target object if constraints are unmet. The **oss_prompt_for_unverifiable_queries** focuses on selecting the single image that most clearly and completely depicts the target object from a 360-degree scan, while the **oss_prompt_for_verifiable_queries** incorporates a three-step reasoning strategy, identifying the earliest image containing the anchor and then limiting the search space for target localization accordingly. The **relation_parsing_prompt** is used to infer the spatial relation (e.g., up, down, near, far, between) between the target and anchor objects from the query. The **sars_choose_target_prompt** performs target selection under a 360-degree rotation by evaluating multiple views and returning the most confident match. The **compare_prompt** determines whether two images captured from the same pose show the target object at the same position, supporting consistency checks. The **fallback_prompt** implements a robust two-step procedure: locating a query-matching image if available, or falling back to the clearest image showing the object class. The **get_good_view_prompt** is used to retrieve up to four images that provide the best views of a reference object based on a reference image with a bounding box. Finally, the **bboxchoose_prompt** refines object selection by identifying the most probable target object among multiple candidate boxes, integrating query content and spatial descriptions. Together, these prompts provide a structured, interpretable, and modular interface for vision-language agents to perform complex multi-view spatial reasoning and object grounding tasks. The textbflimit_prompt guides the VLM to assess whether the target object is overly large or partially occluded, serving as a prior for filtering unreliable candidate point clouds. =@p1.0@

---

memory_retrieval_prompt_for_unverifiable_queries

You are an intelligent assistant proficient in analyzing images. Given a series of indoor room images from a video, you need to analyze these images and select the best 3 images. Each image has an ID in the upper left corner indicating its sequence in the video. Multiple images may be combined and displayed together to save the place. The anchor object is {anchor_class}. If there are some images that are very similar, only select the clearest one to participate in the further selection process. Select the best 3 images from the remaining images according to the following rule: Rule 1: Select those images from the remaining ones that can clearly display the anchor object until the total number of selected images reaches 3. Please reply in json format, including "reasoning" and "selected_image_ids":
{
"reasoning": "Your reasoning process", // Your thinking process regarding the selection task
"selected_image_ids": ["00045", "00002", "..."], // A list of the IDs of the best 3 images selected according to the rules. Note that the returned IDs should be in the form of "00045", not "00045.color", and do not add any suffix after the numbers.
"unique_question": 6 // This is an independent question. Regardless of any other factors, only look for which image among all those provided captures the object {targetclass} most clearly. If none is found, return -1.
}
Now start the task: There are {num_view_selections} images for you to select from.

---

memory_retrieval_prompt_for_verifiable_queries

Imagine that you are in a room and tasked with finding a specific object. You already know the query content: {query}, the anchor object class: {anchorclass}, and the target object class: {targetclass}. The provided images are obtained by extracting frames from a video. Your task is to analyze these images to locate the target object described in the query.

You will receive multiple images, each with an ID marked in the upper left corner to indicate its order in the video. Adjacent images have adjacent IDs. Note that, to save space, multiple images may be combined and displayed together. You will also be given the query statement and a parsed version specifying the target object class and conditions.

Your task is divided into two main steps:

Step 1: Based on the query and associated conditions, determine whether any of the provided images contain the target object that satisfies the requirements. If found, return the corresponding image ID; if not, return -1.

Step 2: If no matching image is found in Step 1, ignore the query content and examine all images to see if any clearly capture an object of class {targetclass}. If such an image exists, return its ID; otherwise, return -1.

Please note that the query statement and conditions may not be fully satisfied in a single image, and they may also contain inaccuracies. Your goal is to find the object that most likely satisfies the query. If multiple candidates exist, choose the one you are most confident about.

Your response should be a JSON object containing the following fields:

{
"reasoning": "Your reasoning process", // Explain how you judged and located the target object. If cross-image reasoning is used, specify which images were involved and how.
"find_or_not": true, // Return true if a suitable image matching the query is found, otherwise return false.
"target_image_id": 4, // Return the image ID that best satisfies the query and conditions. If none found, return -1.

"anchor_image_id": 6, // Return the ID of the image where the anchor object is most clearly visible.
"extended_description": "The target object is a red box located in the lower left corner of the image.", // Describe the target object in the selected image, focusing on color and position.
"unique_question": 6 // This is an independent question. Regardless of other factors, select the image that most clearly captures an object of class {targetclass}. If none, return -1.
}

Now start the task:
There are {num_view_selections} images for your reference.
The following are the conditions for the target object: {condition}

---

oss_prompt_for_unverifiable_queries

Imagine that you are in a room and tasked with finding a specific object. You already know the query content: {query}, the anchor object class: {anchorclass}, and the target object class: {targetclass}. The provided images are frames extracted from a video in which the camera performs a full 360-degree rotation around a specific point. Your task is to analyze these images to locate the target object described in the query.

You will receive multiple images, each with an ID marked in the upper left corner indicating its sequence in the video. Adjacent images have adjacent IDs. To save space, multiple images may be combined and displayed together. Additionally, you will be provided with the query statement and its parsed version, which specify the target class and grounding conditions.

Your goal is to find the image that most clearly and completely captures the target object described by the query. The conditions may not be fully accurate or verifiable from a single image, so the correct object may not satisfy all of them. Try your best to identify the object that most likely meets the conditions. If multiple candidates appear correct, choose the one you are most confident about.

While checking each image, consider different views throughout the 360-degree rotation. If you find the target object in an image, also examine whether other images capture the same object more clearly or completely, and return the best one. Your answer should be based on the image where the target object is most clearly and completely visible.

Please reply in JSON format, structured as follows:

{
"reasoning": "Your reasoning process", // Explain the process of how you identified and located the target object. If reasoning across multiple images is used, explain which images were referenced and how.
"target_image_id": 1, // Replace with the actual image ID (only one) that most clearly captures the target object.
"reference_image_ids": [1, 2, ...], // A list of image IDs that also contain the target object and helped in reasoning.
"extended_description": "The target object is a red box. It has a black stripe in the middle.", // Describe the target object's appearance based on the selected image. Color and features only; do not include position.
"extended_description_withposition": "The target object is a red box located in the lower left corner of the image." // Describe the target object with both appearance and spatial position in the image.
}

Now start the task:
There are {num_view_selections} images for your reference.
Here is the condition for the target object: {condition}

---

oss_prompt_for_verifiable_queries

Imagine that you are in a room with the task of finding specific objects. You already know the query content: {query}, the anchor object category: {anchorclass}, and the target object category: {targetclass}. The provided images are extracted frames from a video that rotates around a certain point. Each image is marked with an ID in the top-left corner to indicate its sequence in the video. Adjacent images have adjacent IDs. For space efficiency, multiple images may be combined and displayed together.

You will also receive a parsed version of the query, which clearly defines the target object category, anchor object category, and grounding conditions.

Your task consists of the following three steps:

Step 1: Based on the anchor object category, determine whether any of the provided images clearly capture the anchor object. If no such image is found, return -1 directly.

Step 2: If Step 1 is successful, return the smallest image ID (denoted as min_ID) among the images that clearly capture the anchor object.

Step 3: Among the images with IDs from 0 to min_ID, try to find an image that clearly captures the target object and satisfies the query content and conditions. If such an image is found, return its ID; otherwise, return -1.

Note: The query statement and conditions may not be perfectly accurate or fully visible in a single image. Try your best to locate the object that is most likely to match these conditions. If multiple objects are plausible, select the one you are most confident about.

Here is an example: In Step 1, images 12, 13, 14, and 15 all clearly capture the anchor object, so Step 2 yields min_ID = 12. In Step 3, no image from ID 0 to 12 meets the query requirements, so target_image_id = -1.

Please reply in JSON format as follows:

```
{
"reasoning": "Your reasoning process", // Explain the reasoning process across all three steps. If cross-image reasoning is involved, specify which images were used and how.
"anchor_image_id": 12, // Return the smallest image ID that clearly captures the anchor object. If none is found, return -1.
"target_image_id": 4, // If anchor_image_id = -1, then return -1 directly. Otherwise, return the image ID ( anchor_image_id) that best satisfies the query. If none found, return -1.
"extended_description": "The target object is a red box located in the lower-left corner of the image.", // Describe the target object in the image with ID = target_image_id. If target_image_id = -1, return None.
"unique_question": 6 // This is an independent question. Regardless of other factors, return the ID of the image that most clearly captures an object of class {targetclass}. If none found, return -1.
}
```

Now start the task:
There are {num_view_selections} images for your reference.
Here are the conditions for the target object: {condition}

relation_parsing_prompt

You are an agent who is highly skilled at analyzing spatial relationships. You are given a query: {query}, a target object: {classtarget1}, and an anchor object: {anchorclass}. Your task is to determine the spatial relationship of the target object relative to the anchor object based on the query content.

The possible spatial relationships are defined as follows:

- **up**: the target object is above the anchor object // the target object is lying on the anchor object // the target object is on top of the anchor object. - **down**: the target object is below the anchor object // the target object is supporting the anchor object // the anchor object is on top of the target object. - **near**:

the target object is close to the anchor object. - **far**: the target object is far from the anchor object. - **between**: the target object is between multiple anchor objects.

Please reply in JSON format with one key, "reasoning", indicating the spatial relationship you determine:

{
"reasoning": "up" // Return the spatial relationship type (up, down, near, far, or between) that best describes the position of the target object relative to the anchor object.
}

Now start the task.

---

sars_choose_target_prompt

Imagine you're in a room tasked with finding a specific object. You already know the anchor object class: {anchorclass}, the target object class: {targetclass}, and the query the target object should match: {query}. The provided images are captured during a 360-degree rotation around the anchor object.

You are given a sequence of indoor-scanning video frames and a query describing a target object in the scene. Your task is to analyze the images and locate the target object according to the query content.

Each image is annotated with an ID in the top-left corner indicating its sequential position in the video. Adjacent images have adjacent IDs. For space efficiency, multiple images may be combined and displayed together. You are also provided with a parsed version of the query, which lists the conditions that the target object should satisfy.

After filtering and comparison, your goal is to identify the image ID that contains the target object most clearly based on the query and conditions. Note that these conditions may not be fully observable in a single image and might be imprecise. The correct object may not meet all conditions. Try to find the object that most likely satisfies them. If multiple candidates seem plausible, choose the one you are most confident about. If no object meets the query criteria, make your best guess. Usually, the target object appears in several images—return the one where it is captured most clearly and completely.

Please reply in JSON format with the following structure:

{
"reasoning": "Your reasoning process", // Explain how you identified and located the target object. If you used multiple images, describe which ones and how they contributed to your decision.
"target_image_id": 1, // Replace with the actual image ID that most clearly shows the target object. Only one ID should be provided.
"reference_image_ids": [1, 2, ...], // A list of other image IDs that also helped confirm the target object's identity.
"extended_description": "The target object is a red-colored box. It has a black stripe across the middle.", // Describe the target object's color and notable features. No need to mention its position.
"extended_description_withposition": "The target object is a red-colored box located in the lower left corner of the image." // Describe both appearance and position of the object in the selected image.
}

Now start the task:
There are {num_view_selections} images for your reference.
Here is the condition for the target object: {condition}

---

compare_prompt

You are an intelligent assistant who is extremely proficient in examining images. You already know the target object category: {target_class}. Now I will provide you with two images. You need to determine whether the target objects captured in these two images are in the exact same position. Since these two images are taken from the same pose, you only need to check whether the target objects are in the same position within the images.

For example, if the target object is a table and you can clearly see that the table is located in the middle of both images, then the target objects captured in these two images are considered to be in the same position.

Please reply in JSON format with two keys: "reasoning" and "images_same_or_not":

```
{
"reasoning": "Your reasons", // Explain the basis for your judgment on whether the target objects captured in these two images are in the same position.
"images_same_or_not": true // It should be true if you think the target objects captured in the two images are in the same position. If you find that the positions of the target objects captured in the two images are different, or if the target object is captured in the first image but not in the second, then it should be false.
}
```

fallback_prompt

Imagine you are in a room tasked with finding a specific object. You already know the query content: {query}, and the target object category: {targetclass}. The images provided to you are frames extracted from a video that rotates around a particular point. Each image is marked with an ID in the top-left corner to indicate its sequence in the video, and adjacent images have consecutive IDs. For space efficiency, multiple images may be combined and displayed together.

Your task consists of two steps:

Step 1: Locate an image that contains the target object which satisfies the query statement and its associated conditions. The image must clearly and completely capture the target object. If such an image is found, return its ID and skip Step 2.

Step 2: If no image meets the query-based requirements, ignore the query and check all provided images. Identify an image that clearly captures the object of category {targetclass}. If such an image is found, return its ID. If none are found, return -1.

Please reply in JSON format with the following structure:

```
{
"reasoning": "Your reasoning process", // Explain the reasoning behind both steps of your decision-making process.
"match_query_id": 12, // Return the image ID that satisfies Step 1. If no image matches the query, return -1.
"object_image_id": 4, // If Step 1 is successful, return -1 here. Otherwise, return the ID of the image that clearly captures the object in Step 2. If not found, return -1.
"extended_description": "The target object is a red box located in the lower-left corner of the image." // Provide a brief description of the target object as seen in the selected image. Focus on visual features such as color and location within the image.
}
```

Now start the task:
There are {num_view_selections} images for your reference.

get_good_view_prompt

You are an excellent image analysis expert. I will now provide you with several images, each marked with an ID in the upper left corner. These images are captured by rotating around a target object {target} that is framed with a green bounding box in the reference image. The reference image is also provided, and it contains the target object {target} enclosed by a green box, with the word "refer" shown in red in the upper left corner.

Your task is to determine which three (at most four) of the provided images capture the target object from the reference image most clearly and completely. Please note that, for layout efficiency, multiple images may be displayed together in a single composite image.

Your response should be in JSON format, containing the following fields:

{
"reasoning_process": "Your reasoning process", // Explain how you select the images that best capture the target object framed in the reference image.
"image_ids": [2, 4, 5, 7] // Replace with the actual image IDs. Return up to four IDs corresponding to the images that, in your opinion, capture the target object most clearly and completely.
}

Now start the task:
There are {num_images} candidate images and one reference image for you to choose from.

bboxchoose_prompt

Great! Here is the detailed version of the picture you've selected. There are {num_candidate_bboxes} candidate objects shown in the picture. I have annotated an object ID at the center of each object with white text on a black background. You already know the query content: {query}, the anchor object: {anchorclass}, and the target object: {classtarget}. In addition, you will be provided with an extended description: {description}, which includes the position of the target object in the picture.

Your task consists of two main steps:

Step 1: The candidate objects shown in the picture are not necessarily all of the target class {classtarget}. You must first determine which of them belong to the class {classtarget}.

Step 2: Among the identified candidate objects of class {classtarget}, select the one that best matches both the query content and the extended description (including position).

Please reply in JSON format with two fields:

{
"reasoning": "Your reasoning processing", // Describe your full reasoning process in three parts: (1) how you identified candidate objects of the target class; (2) how you verified them against the extended description; and (3) how you selected the final object ID.
"object_id": 0 // The object ID you select. Always provide one object ID from the picture that you are most confident about, even if you think the correct object might not be present.
}

Now start the task: There are {num_candidate_bboxes} candidate objects in the image.

limit_prompt

Great! Now you will perform an expert judgment on the visibility of a target object in the provided image.

You already know the target object category: {targetclass}. You will be shown one image containing this object class.

Your task consists of two main steps:

Step 1: Some object categories, such as beds, sofas, closets, cabinets, shelves, etc., are considered inherently large. If the target object belongs to this group of large categories, directly return `"limit": true` without proceeding to the next step.

Step 2: If the target class is not considered large, examine the image and determine whether the target object appears to be fully captured. If you believe the object is incomplete or partially outside the frame, return `"limit": true`; otherwise, return `"limit": false`.

Please reply in JSON format with two fields:

```
{
"reasoning": "Your reasoning process", // Describe your reasoning clearly: (1) whether the category is considered large, and (2) if not, how you judged the completeness of the object in the image.
"limit": false // Return true only if the object is large, or if it is not large but appears incomplete in the image.
}
```

Now start the task: You are given one image and the target object category: {targetclass}.

G Cost calculation for methods

For both the baseline methods and our proposed MCG approach, the robot's initial camera pose is assumed to be at the center of the room (see the main text for the formal definition of this key assumption). For MCG, the full camera trajectory starts from the initial pose and follows a sequence of new poses generated by the MCG pipeline. The cost of the entire trajectory is computed according to the evaluation metrics defined in the main paper. For the WG and CRG baselines, all images are pre-captured and sequentially indexed. We first identify the image whose pose is closest to the initial camera pose and denote its index as n. The camera trajectory then starts from the initial pose and proceeds through the poses of images with indices $n, n+1, n+2, \ldots$, wrapping around from the last index back to 1 as needed, and ending at index $n-1$. The cost is computed based on the same evaluation procedure. For the MOG baseline, which only utilizes memory images, the camera trajectory consists of only two poses: the initial pose and the pose of the target image. Its cost is similarly computed using the defined metrics.

H More results

H.1 Inference time and success rate

H.2 Rendered vs. real images in memory

In previous experiments, both the memory and exploration images used by our system were rendered images. However, the effectiveness of vision-language models (VLMs) on rendered images—in terms of recognition, reasoning, and analysis—remains unclear. In this section, we explore the impact of replacing memory images with real-world images. Specifically, we randomly sample 50 instances from a pool of 250 and observe the final localization results. Experimental findings indicate that using rendered images in memory does not significantly affect the overall grounding accuracy. However, this should not be interpreted as rendered images outperforming real ones. In repeated experiments, we observed slight fluctuations in accuracy between the two settings. These results suggest that using rendered images for memory introduces negligible impact on overall performance.

Table 7: Comparison between using rendered and real images in memory.

Version	@0.25	A_c	M_c
w. rendering	28	1.74	2.08
w/o. rendering	24	1.62	1.96

H.3 Error case illustration

In this section, we present concrete failure cases of key modules in the MCG framework.

VLMs failure in memory retrieval Owing to the inherent limitations of the VLM, it may fail to correctly identify the anchor object in image sequences from memory, thereby causing a cascade of errors in the subsequent grounding pipeline

VLMs failure in target image retrieval Relational queries involving horizontal spatial reasoning (e.g., "select the chair closest to the table") impose higher demands on the inference capability of vision-language models (VLMs). Such relationships require the model to make fine-grained comparisons based on relative spatial distances rather than absolute object properties. The difficulty is further amplified in cluttered scenes where multiple distractor objects are present, and the distance difference between the correct target and nearby alternatives is minimal. In these cases, the VLM is more prone to incorrect selections due to the subtlety of the distinction required.

Failure in SARS The method used in SARS for acquiring new observation viewpoints can result in limited viewing angles, which may prevent the target object from entering the field of view—especially in cases where the object is positioned too low. Moreover, since we rely on rendered images, the presence of rendering noise further exacerbates the issue. In certain viewpoints, the rendered images may contain large blank or missing regions, making them unusable for grounding.

Failure in SAM and projection Although the SAM segmentation model demonstrates strong overall performance, it still introduces a considerable amount of noise. During our experiments, we observed that SAM frequently includes pixels unrelated to the target object in the final segmentation mask. This over-segmentation adversely affects the subsequent 3D projection process, ultimately leading to reduced accuracy in the localization of the 3D bounding box. In addition, since our experiments are conducted on rendered images, the resulting RGB-D data often contain missing or incomplete regions, which further impact the precision of the 3D bounding box estimation. Although we have attempted to denoise the rendered images as much as possible—especially by removing abrupt pixel changes in the depth maps—some residual noise and artifacts may still persist, continuing to pose challenges to accurate 3D localization.

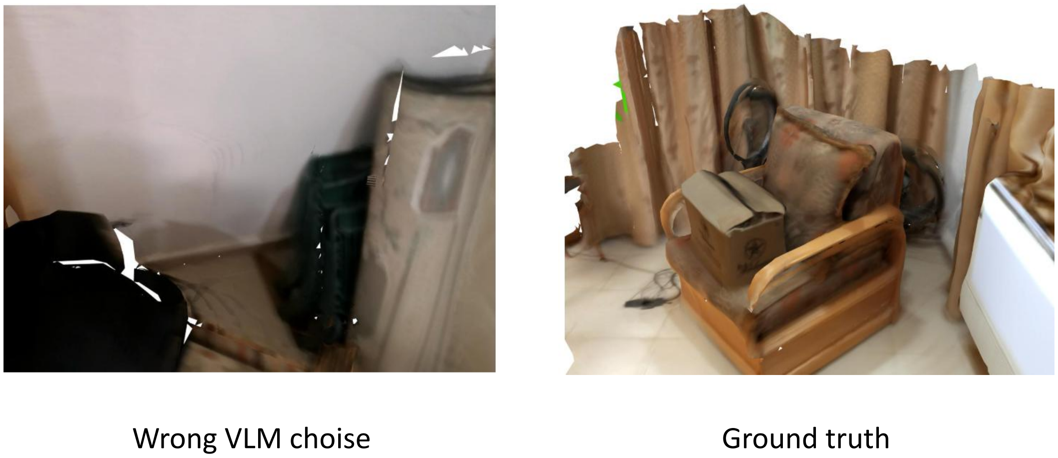


Figure 5: **VLMs failure in memory retrieval, the anchor object is box.**

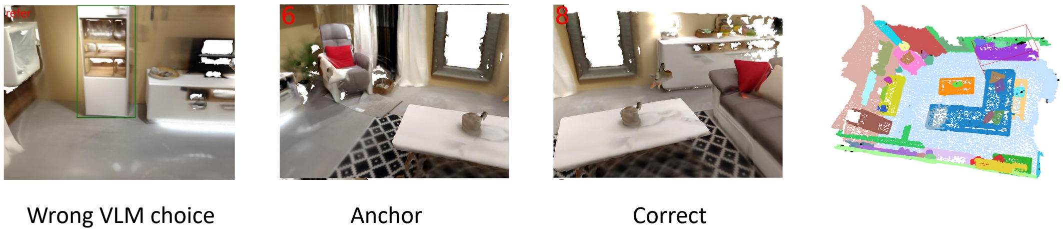


Figure 6: **VLMs failure in target image retrieval, the user query is to choose the cabinet near the box.**

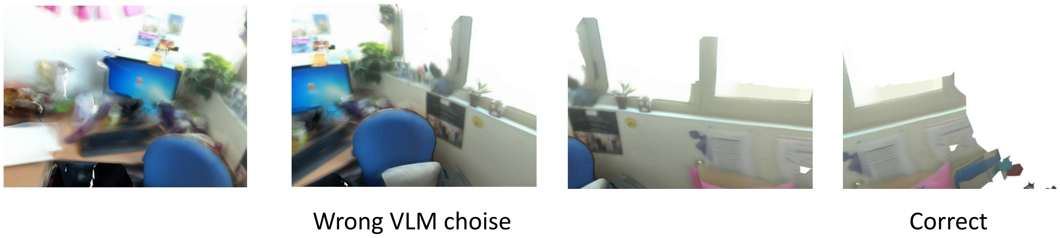


Figure 7: **Failure in SARS, the user query is to find the cushion that is farthest from the pc.**

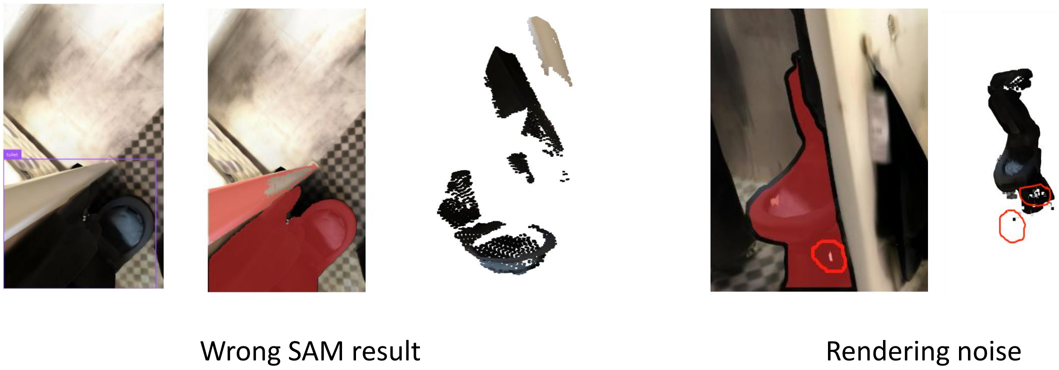
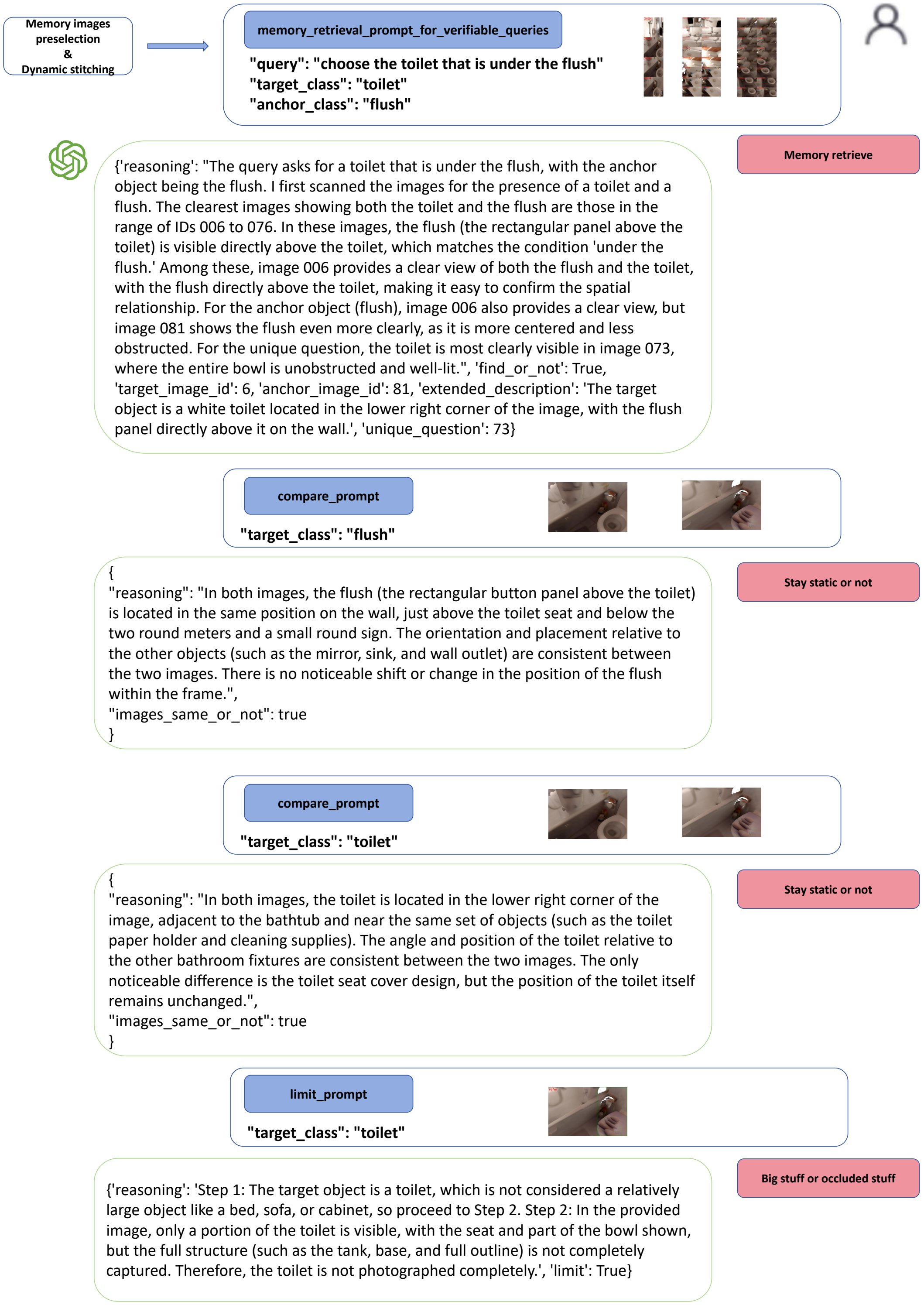


Figure 8: **Failure in SAM and projection.**

H.4 Full demo

In this section, we present a representative and structurally concise example to intuitively illustrate how MCG effectively leverages memory information for efficient and accurate target localization in dynamic environments. The example highlights the central role played by the vision-language model (VLM) throughout the entire execution process. We provide a detailed depiction of the VLM's reasoning at each step, demonstrating how it progressively converges on the target object through multi-round perception and decision-making, thereby showcasing its capabilities in semantic understanding and spatial reasoning.


Memory images preselection & Dynamic stitching
memory_retrieval_prompt_for_verifiable_queries

"query": "choose the toilet that is under the flush"
"target_class": "toilet"
"anchor_class": "flush"

{'reasoning': "The query asks for a toilet that is under the flush, with the anchor object being the flush. I first scanned the images for the presence of a toilet and a flush. The clearest images showing both the toilet and the flush are those in the range of IDs 006 to 076. In these images, the flush (the rectangular panel above the toilet) is visible directly above the toilet, which matches the condition 'under the flush.' Among these, image 006 provides a clear view of both the flush and the toilet, with the flush directly above the toilet, making it easy to confirm the spatial relationship. For the anchor object (flush), image 006 also provides a clear view, but image 081 shows the flush even more clearly, as it is more centered and less obstructed. For the unique question, the toilet is most clearly visible in image 073, where the entire bowl is unobstructed and well-lit.", 'find_or_not': True, 'target_image_id': 6, 'anchor_image_id': 81, 'extended_description': 'The target object is a white toilet located in the lower right corner of the image, with the flush panel directly above it on the wall.', 'unique_question': 73}

"target_class": "flush"

{
"reasoning": "In both images, the flush (the rectangular button panel above the toilet) is located in the same position on the wall, just above the toilet seat and below the two round meters and a small round sign. The orientation and placement relative to the other objects (such as the mirror, sink, and wall outlet) are consistent between the two images. There is no noticeable shift or change in the position of the flush within the frame.",
"images_same_or_not": true
}

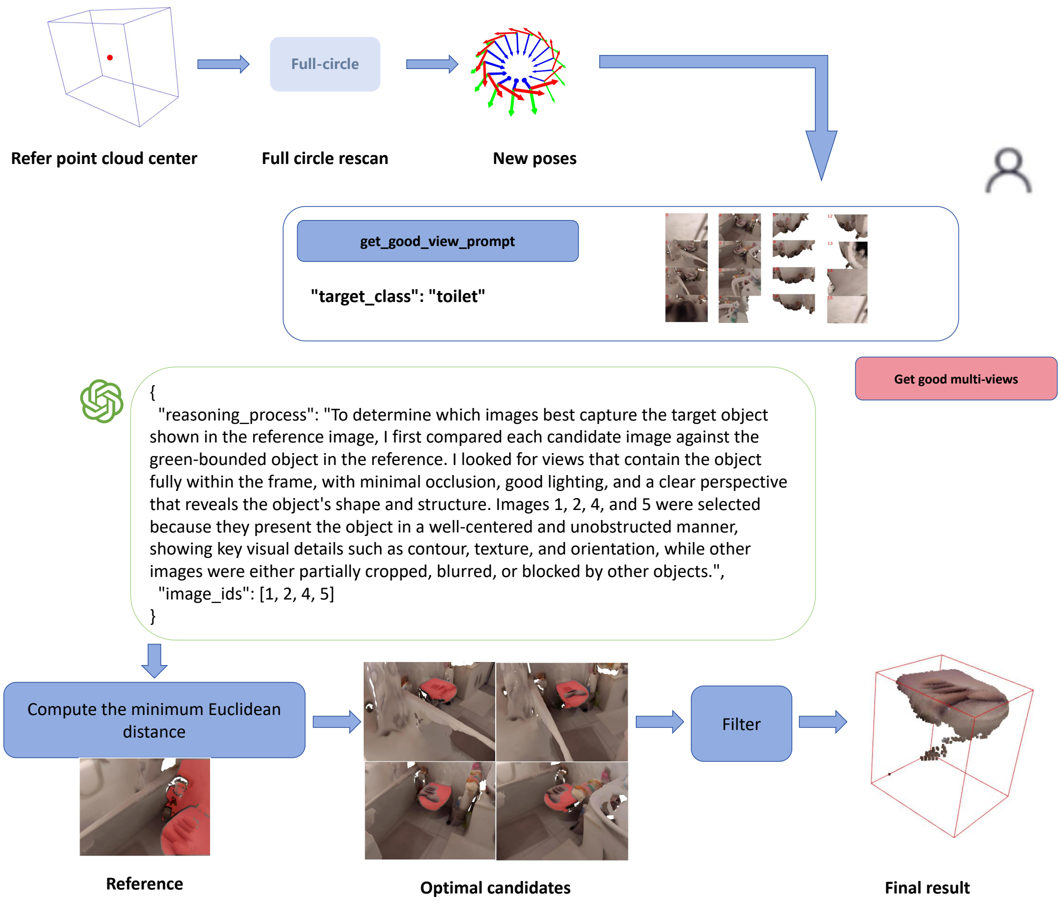
"target_class": "toilet"

{
"reasoning": "In both images, the toilet is located in the lower right corner of the image, adjacent to the bathtub and near the same set of objects (such as the toilet paper holder and cleaning supplies). The angle and position of the toilet relative to the other bathroom fixtures are consistent between the two images. The only noticeable difference is the toilet seat cover design, but the position of the toilet itself remains unchanged.",
"images_same_or_not": true
}

"target_class": "toilet"

{'reasoning': 'Step 1: The target object is a toilet, which is not considered a relatively large object like a bed, sofa, or cabinet, so proceed to Step 2. Step 2: In the provided image, only a portion of the toilet is visible, with the seat and part of the bowl shown, but the full structure (such as the tank, base, and full outline) is not completely captured. Therefore, the toilet is not photographed completely.', 'limit': True}

Figure 9: **Case of the MCG grounding part-1.**



```
{
  "reasoning_process": "To determine which images best capture the target object shown in the reference image, I first compared each candidate image against the green-bounded object in the reference. I looked for views that contain the object fully within the frame, with minimal occlusion, good lighting, and a clear perspective that reveals the object's shape and structure. Images 1, 2, 4, and 5 were selected because they present the object in a well-centered and unobstructed manner, showing key visual details such as contour, texture, and orientation, while other images were either partially cropped, blurred, or blocked by other objects.",
  "image_ids": [1, 2, 4, 5]
}
```

Figure 10: **Case of the MCG grounding part-2.**