MO-DDN: A Coarse-to-Fine Attribute-based Exploration Agent for Multi-object Demand-driven Navigation

Hongcheng Wang[1,3*]   **Peiqi Liu**[2*]
Wenzhe Cai[4]   **Mingdong Wu** [1,3]   **Zhengyu Qian** [2]   **Hao Dong** [1,3†]
[1]CFCS, School of CS, PKU   [2]School of EECS, PKU   [3]PKU-Agibot Lab
[4]School of Automation, Southeast University

Abstract

The process of satisfying daily demands is a fundamental aspect of humans' daily lives. With the advancement of embodied AI, robots are increasingly capable of satisfying human demands. Demand-driven navigation (DDN) is a task in which an agent must locate an object to satisfy a specified demand instruction, such as "I am thirsty." The previous study typically assumes that each demand instruction requires only one object to be fulfilled and does not consider individual preferences. However, the realistic human demand may involve multiple objects. In this paper, we introduce the Multi-object Demand-driven Navigation (MO-DDN) benchmark, which addresses these nuanced aspects, including multi-object search and personal preferences, thus making the MO-DDN task more reflective of real-life scenarios compared to DDN. Building upon previous work, we employ the concept of "attribute" to tackle this new task. However, instead of solely relying on attribute features in an end-to-end manner like DDN, we propose a modular method that involves constructing a coarse-to-fine attribute-based exploration agent (C2FAgent). Our experimental results illustrate that this coarse-to-fine exploration strategy capitalizes on the advantages of attributes at various decision-making levels, resulting in superior performance compared to baseline methods. Code and video can be found at https://sites.google.com/view/moddn.

1 Introduction

In the field of psychology, the creation of demands directs the motivation for human behavior in the real world, and behavior ultimately leads to the fulfillment of the demand [1–4]. For example, an individual might crave sweet iced tea, prompting them to search for object such as tea, sugar, ice, and water, and then combine them to make sweet iced tea. Recently, with the rapid development of embodied AI and large language models, researchers are interested in using robots to meet various human demands [5–13]. Demand-driven Navigation (DDN) [14] is a variant of ObjectGoal Navigation (ON) [15–21], which requires an agent to find an object that satisfies a given demand instruction. For example, when a user gives the agent an instruction such as "I am thirsty," the robot has to search the entire environment for objects such as water, tea, coffee, etc., depending on what is available within the environment.

In previous work [14], demand instructions often exhibit a low level of complexity, lack consideration for user preferences, and typically require only one object to fulfill each instruction. However,

---

*Equal contribution.
†Corresponding author.

38th Conference on Neural Information Processing Systems (NeurIPS 2024).
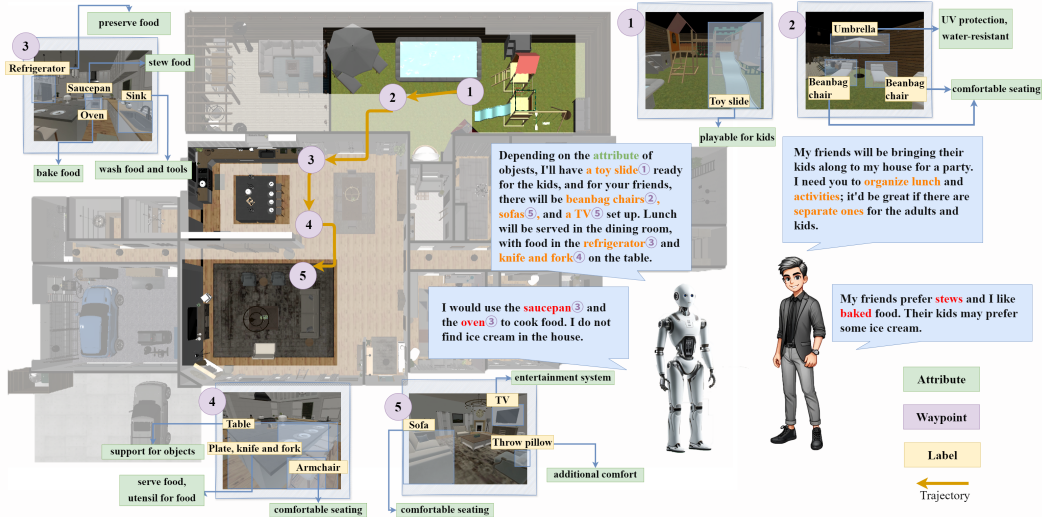
Figure 1: **An Example of Multi-object Demand-driven Navigation.** A user plans to host a party at his new house and outlines some basic demands (highlighted in orange), along with specific preferences for different individuals (highlighted in red). The agent parses these demands and locates multiple objects in various locations in the scene to fulfill them. Despite not meeting the preferred "ice cream" demand, the agent successfully addresses basic demands, such as organizing lunch.

real-world situations frequently involve more intricate instructions, necessitating the coordination of multiple objects to satisfy a demand. For example, in the BEHAVIOR-1K [22], daily tasks often involve the interaction of multiple objects. Furthermore, individual users may have distinct preferences. For instance, when presented with the basic demand "I am thirsty," one user may prefer sparkling water while another may favor juice. However, satisfying preferred demands is obviously more difficult than satisfying basic demands, so the agent needs to be flexible in prioritizing goals depending on both the situations of users and environments.

We introduce a new benchmark, Multi-object Demand-driven Navigation (MO-DDN), in which demand instructions are comprised of basic and preferred demand components. An agent must find a combination of objects (we call it a "**solution**" later) that satisfies the demand. The agent should satisfy the user's preferred demands as much as the situation allows. We use GPT-4 [3] to automatically generate and modify tasks and perform manual checks. Our settings are similar to Multi-object Navigation (MON) [23–25], but we use a demand instruction to combine potentially multiple objects in a form that is consistent with common sense and personal preferences of humans rather than a list of object categories in MON. The number and category of objects in a solution are not known in advance but require an agent to reason on its own. Moreover, solutions that satisfy the same demand may have a different number of objects. We argue that MO-DDN has similar advantages over MON as DDN has over ON.

We consider MO-DDN to be a crucial preliminary step in task planning. Today, with sufficient meta-information provided (*e.g.*, position, state, and category of all objects in the scene), large language models can provide satisfactory task planning results [26–29, 5]. **However, maintaining and providing accurate meta-information can be challenging and inconvenient in real life.** The user's interactions with the scene change the meta-information frequently. Moreover, since the user is not necessarily omniscient about the scene, especially in unfamiliar scenes, it is possible that if the user parses the demand himself and directly provides the desired objects, these desired objects may not exist in the scene. Even if the user can provide comprehensive meta-information, delivering the meta-information to the agent is time-consuming and inconvenient for the user. Therefore, the purpose of MO-DDN is to provide timely and accurate meta-information depending on **a convenient demand instruction** in natural language for later task planning or, more later, object manipulation.

---

[3] we use gpt-4-0125-preview API in this paper here and later, unless otherwise noted.

2

In the recently ObjectGoal Navigation work, benefiting from the development of Vision Language Models (VLMs) [30–36] and Large Language Models (LLMs) [37–42], some ObjectGoal Navigation methods return to a modular manner [43, 44, 17, 45–50] rather than an end-to-end manner [19, 51–58]. While maintaining the core concept of the "attribute" from the end-to-end agent of the previous work [14], we modify the training of the attribute model to make it applicable to multi-object search, and propose a coarse-to-fine attribute-based exploration modular agent, **C2FAgent**. In the coarse exploration phase, similar to the modular methods in ObjectGoal Navigation, we use a depth camera to reconstruct the environment point clouds and an object detection module, GLEE [31], to label the objects in the point clouds. Additionally, the point clouds are compressed and segmented into several 2D rectangular blocks, and thus each detected object belongs to a block. For each block, we calculate the similarity of the objects' attribute features and the instruction's attribute features as a metric for choosing a waypoint. We calculate basic demand similarity scores and preferred demand similarity scores separately and select blocks according to a weighted sum of the two scores. In the fine exploration phase (*i.e.*, when the agent arrives at the waypoint), we train an end-to-end attribute exploration module to identify and report the target object to compose the solution.

The experimental results show that our proposed method outperforms the baselines, and the ablation study shows that the attribute model improves exploration efficiency in different phases. We argue that this coarse-to-fine design allows for the incorporation of prior knowledge from external foundation models in the coarse exploration phase and task-relevant world-grounding exploration in the fine exploration phase. Moreover, the ablation study on the weighted sum of the basic and preferred similarity scores demonstrates that increasing the preferred weights allows the agent to prioritize searching for the preferred solution and increasing the basic weights allows the agent to prioritize searching for the basic solutions. Therefore, users can adjust the weights to influence the agent's behavior freely, which is a flexible way to handle personal preferences.

Our main contributions are listed as follows:

- We propose a new benchmark, MO-DDN, which considers multi-object combinations as solutions and more complex and diverse demand instructions. MO-DDN can be regarded as a crucial preliminary step in task planning.

- We extend the training process of the attribute model, enabling the attribute features to work well in a multi-object setting. Based on the new version of attribute features, We design a coarse-to-fine attribute-based exploration agent, C2FAgent, for this benchmark, allowing the attribute features to play an important role in different exploration phases.

- The experimental results show that the attribute features do improve the efficiency of exploration, and the experimental results substantially surpass the baselines. Ablation study shows that attribute-based exploration is more efficient than frontier-based exploration [59] and LLM-based waypoint selection.

2 Related Work

2.1 Visual Navigation

Goal Description   In general terms, visual navigation involves the continuous generation of actions based on RGB-D and GPS+Compass inputs until a specified objective is reached [60]. Visual navigation tasks vary in their goal description, such as step-by-step instructions in Vision-Language Navigation (VLN) [61–68], audio in Audio-visual Navigation [69–74], object categories in Object-Goal Navigation (ON) [15–21], object category lists in MultiObject Navigation (MON) [43, 23–25], and demand instructions in Demand-driven Navigation (DDN) [14]. Our proposed benchmark, MO-DDN, can be viewed as a multi-object version of DDN. Although, like DDN, MO-DDN's inputs are demand instructions, MO-DDN's solutions are multi-object rather than single-object, as in DDN.

Previous Method Overview   In ON, with the rise of object detection models [31, 30, 75, 76], object segmentation models [77–80], and large language models, modular methods [43, 44, 17, 45–50] are gradually being developed. They greatly improve navigation efficiency and success rate by building semantic maps and navigable maps and then planning paths on the maps. Meanwhile, end-to-end methods [19, 51–58] focus more on learning associations in object-object and object-scene to help reason about the potential location of target objects. In MON, previous work has focused on

3

studying how to quickly construct [25], memorize [24, 43], and use semantic maps [81] or implicit representations [23] of scenes. In DDN, the concept of attributes is introduced as a way of expressing what an object shows when it fulfills a demand, *e.g.*, quenching thirst is an attribute of water in the context of demand "I am thirsty". In this paper, we extend the concept of attributes and propose a new method to train the attribute model. We combine the advantages of modular and end-to-end methods and propose a coarse-to-fine attribute-based exploration modular agent.

2.2 Foundation Large Model in Embodied Task

Foundation Large Models refer to self-supervised pre-trained models trained on large Internet-scale datasets, which have demonstrated promising capabilities across various embodied tasks. Large language models (LLMs) such as GPT-4 [82], LLaMA [83, 40], and Gemma [84] exhibit performance comparable to humans in task planning [26–29, 85], common sense reasoning [86] and question answering [87–90, 34, 91]. Furthermore, researchers use LLMs to synthesize task datasets [14]. The CLIP model [92], which employs contrastive learning with image and text pairs, showcases strong semantic extraction abilities in navigation [14, 93, 21, 68, 94]. Visual Language Models offer understanding of images such as image descriptions [95–98], object detection [31, 30, 75, 76], and object segmentation [77–80] for downstream embodied tasks. This paper uses LLMs, specifically GPT-4, to generate task instructions, solutions, and attribute examples for attribute learning. Furthermore, we employ GLEE [31], a state-of-the-art object detection model, to detect object categories within the field of view for attribute feature extraction.

3 Multi-object Demand-driven Navigation

Following the basic settings in DDN, let $\mathcal{D}$ denote a set of demand instructions, $\mathcal{S}e$ denotes a set of navigable scenes, and $\mathcal{O}$ denotes a set of object categories. Let $\mathcal{S}o$ denote a set of solutions for demand instructions. An element in $\mathcal{S}o$ (*i.e.*, a solution) is a subset of $\mathcal{O}$. In each episode, similar to the DDN task, an agent is randomly initialized to a location within a mapless environment and receives a demand instruction $DI \in \mathcal{D}$ in natural language. In this benchmark, a demand instruction consists of two parts: one is basic demand instruction DI_b, and the other is preferred demand instruction DI_p, *e.g.*, " I need a comfortable place to play computer games, preferably with good lighting." Each DI has two sets of solutions, *i.e.*, basic solution So_b and preferred solution So_p for DI_b and DI_p, respectively. For example, {desk, soft chair} is an element in So_b for the above example demand instruction, and {desk, soft chair, table lamp} is an element in So_p.

Then, at each time step, the agent should choose an action from MoveAhead, RotateRight, RotateLeft, LookUp, LookDown, Find, and Done. The action Find is similar in MON, which automatically reports the objects in the field of view. To reduce the difficulty of the MO-DDN task and focus it on navigation, if the agent chooses Find, all objects in the field of view with distance below the threshold d_{find} will be recorded in a found list FL, instead of requiring the bounding box of the target object as in DDN. The found list FL is used to calculate the basic and preferred success rate later. When the agent chooses Done, or the number of choices Find reaches a threshold n_{find}, or the number of steps reaches a threshold n_{step}, the episode ends and the success rate is calculated. For a specific demand instruction DI, We calculate the basic success rate as follows:

$$SR_b = \frac{1}{N} \sum_{i=1}^{N} \max_{s_b \in So_b} \frac{\sum_{o \in FL} \mathbb{1}_{o \in s_b}}{Len(s_b)} \tag{1}$$

where N donates the number of testing episodes, s_b donates a basic solution in the solution set of DI, FL donates the found list that the agent reports, o donates an object category. The preferred success rate SR_p is calculated similarly. To summarize, the success rate of an episode is the value with the highest percentage of satisfaction among all solutions. We also calculate the SPL [60] corresponding to SR_b and SR_p. We generate 300 tasks, encompassing 358 object categories from the HSSD dataset [99]. These tasks are referred to as HSSD's world-grounding tasks, indicating that objects in these tasks are in the HSSD dataset. We generate some language-grounding tasks to train the attribute model afterward. Language-grounding means that the objects in the solutions can be everything that makes sense rather than restricting objects in HSSD. Please see the supplementary material for details about task generation A.1.1, task metrics A.1.2, and task dataset statistics A.1.3.

4

4 Method

4.1 Attribute Model Training

In this section, we describe how to train the MO-DDN's attribute model that extends from the attribute model in DDN. In DDN, the training of attribute features is constrained by the assumption that each instruction requires only one attribute that can be satisfied by a single object. Such attribute features may not work well under multi-object settings. This paper proposes directly mapping demand instructions and object categories into the same attribute feature space. Such a mapping can learn multiple attribute features simultaneously to address multi-object search. Concretely, the core function of the attribute model is to map a demand instruction in $\mathcal{D}$ or an object category in $\mathcal{O}$ to several attribute features in $\mathcal{R}^d$ ($\mathcal{R}$ is the set of real numbers, d is the dimension of the attribute features) that are in the shared attribute feature space. In order to enable the alignment of the attribute features of instructions and objects, we design a discrete codebook and five losses. The alignment means that for a demand instruction and an object in its solution, one of the attribute features of the instruction and one of the attribute features of the object have a high cosine similarity.
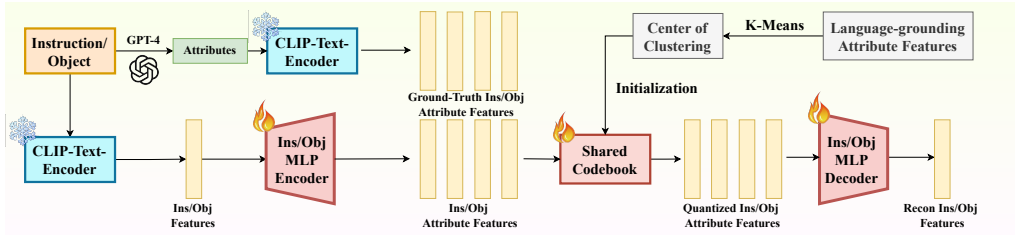
Figure 2: **Attribute Model.** This figure shows the architecture of the attribute model. Instructions and objects share the same model architecture. Instructions and items share the same model architecture. For parameters, they share only the parameters of the shared codebook, while the parameters of the MLP Encoder and Decoder are independent. Only the red with flames modules in the figure will be trained while the blue with snowflakes CLIP model parameters will be frozen.

4.1.1 Codebook and Its Initialization

The reason for using a discrete codebook is similar to VQ-VAE, *i.e.*, different attributes are inherently discrete from each other. Then using a discrete codebook as an attribute feature space is a better way to represent the relationships between attributes than a continuous attribute feature space. The codebook is essentially some feature vectors, and in our experiments, we choose 128 as the number of vectors and 768 as the vector dimension (same with CLIP ViT-L/14's text dimension); thus, the codebook is a 128×768 matrix. we use CLIP-Text-Encoder (ViT-L/14) to encode the language-grounding attributes generated in the supplementary material Sec A.2.1 into attribute features. These attribute features are then clustered using K-means [100] to get 128 clustering centers. The feature vectors of these 128 clustering centers are used to initialize the codebook, making the codebook as a subspace of the CLIP feature space.

4.1.2 Definition of Losses

In our method, there are two attribute models that share the same architecture, the instruction attribute model AM_{ins} and the object attribute model AM_{obj}, shown in Fig. 2. In both models, only MLP Encoder and CLIP Encoder will be used in **C2FAgent**, Codebook and MLP Decoder are only used to train MLP Encoder. We obtain the k_1 instruction attributes and k_2 object attributes (the green square in Fig. 2) by prompting GPT-4 for "what attributes can satisfy this instruction?" and "what attributes does this object have?", respectively. For the five losses used to train the two attribute models, see the pseudo-code 1.

5

Algorithm 1: Losses in Attribute Training

Input: An instruction and an object in the intruction's solution
Output: Five losses
begin

    // Variables in parentheses are full names.
    1. Ins Attributes = AskGPT-4(Instruction);
    2. Obj Attributes = AskGPT-4(Objects) ;
    3. GT-IAF (GT Ins Attribute Features)= CLIP(Ins Attributes) // `shape=`(k_1, d)
    4. GT-OAF (GT Obj Attribute Features)= CLIP(Obj Attributes) // `shape=`(k_2, d)
    5. IF (Ins Feature) = CLIP(Instruction) // `shape=`$(1, d)$
    6. OF (Obj Feature) = CLIP(Objects) // `shape=`$(1, d)$
    7. IAF (Ins Attribute Features) = Ins MLP_Encoder(IF) // `shape=`(k_1, d)
    8. OAF (Obj Attribute Features)= Obj MLP_Encoder(OF) // `shape=`(k_2, d)
    9. Attribute Loss = MSE(GT-IAF, IAF) + MSE(GT-OAF, OAF) ;
    10. Q-IAF (Quantized IAF) = Codebook(IAF)// `shape=`(k_1, d)
    11. Q-OAF (Quantized OAF) = Codebook(OAF)// `shape=`(k_2, d)
    12. Commitment Loss = MSE(IAF, stop-gradient(Q-IAF)) + MSE(OAF, stop-gradient(Q-OAF)) ;
    13. VQ Loss = MSE(Q-IAF, stop-gradient(IAF)) + MSE(Q-OAF, stop-gradient(OAF));
    14. Recon IF = Ins MLP_Decoder(IAF) // `shape=`$(1, d)$
    15. Recon OF = Ins MLP_Decoder(OAF) // `shape=`$(1, d)$
    16. Reconstruction Loss = MSE(Recon IF, IF) + MSE(Recon OF, OF);
    17. Matching Loss = $Minimum_{i=1..k_1, j=1..k_2}(MSE(i_{th}\ \text{IAF}, j_{th}\ \text{OAF}))$

end

The full loss function is shown below:

$$Loss = \lambda_1 \times Attribte\ Loss + \lambda_2 \times VQ\ Loss+$$
$$\lambda_3 \times Commit\ Loss + \lambda_4 \times Recon\ Loss + \lambda_5 \times Matching\ Loss \quad (2)$$

where λ_1 is 2.0, λ_2 is 1.0, λ_3 is 0.25, λ_4 is 1.0, and λ_5 is 1.0. Attribute Loss provides a direct loss for directing the MLP Encoder to learn the projections of IF and OF to IAF and OAF [4]. The next three items VQ Loss, Commitment Loss and Reconstruction Loss can be referred to VQ-VAE Loss [101]. Our motivation in using these three items is to provide indirect constraints that enable the IF and OF to be projected into the shared codebook's feature space. To provide a direct alignment of the attribute features of a given instruction and objects that satisfy the given instruction, we design Matching Loss. An instruction and an object in the instruction's solution theoretically have at least one attribute that matches, *i.e.*, an attribute of the object that necessarily satisfies part of this instruction; otherwise, the object should not be part of the instruction's solution. We consider the most similar pair of attribute features among the k_1 IAF and the k_2 OAF to be a match, and therefore need to reduce the error of this pair of attribute features.

In the end, only the instruction and object MLP Encoders and CLIP-Text-Encoder are used in navigation, and other part like codebook and MLP Decoder are not used. Since this codebook is initialized by CLIP features and the ground-truth attribute features are also encoded by CLIP, the attribute feature space we get can actually be seen as a subspace of the CLIP semantic space. We hope that this design will improve the generalization of attribute features, since the CLIP semantic space shows good generalization and performance on many tasks [94, 93, 102].

4.2 Coarse-to-fine Exploration Agent

In this section, we describe how attribute features work in navigation. The agent switches between coarse and fine exploration phases back and forth until the number of Find executions reaches the upper limit n_{find} or the number of steps reaches the upper limit n_{step}. Fig. 3 illustrates a general overview of the entire navigation policy and Fig. 4 and Fig. 5 show the details of the coarse and fine exploration phase, respectively. In both the coarse and fine exploration phase, we load the parameters of Ins MLP Encoder and Obj MLP Encoder from the attribute training for waypoint (*i.e.*, block) selection and end-to-end fine exploration. We argue that this coarse-to-fine design allows for the incorporation of prior knowledge from external foundation large models in the coarse exploration phase and task-relevant world-grounding exploration in the fine exploration phase.

---

[4]The abbreviations in this section are inherited from the pseudo-code 1.

6
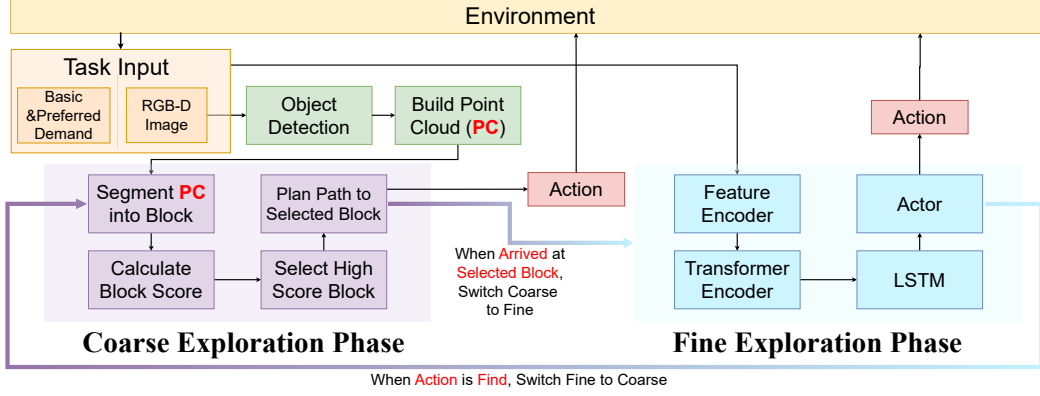
Figure 3: **Navigation Policy.** The agent continuously switches between a coarse exploration phase and a fine exploration phase until the Find count limit n_{find} is reached or the total number of steps n_{step} is reached. See Sec. 4.2.1 and Sec. 4.2.2 for details about the two phases. In each timestep, the GLEE model is used to identify and label objects in the RGB and project them to the point cloud.
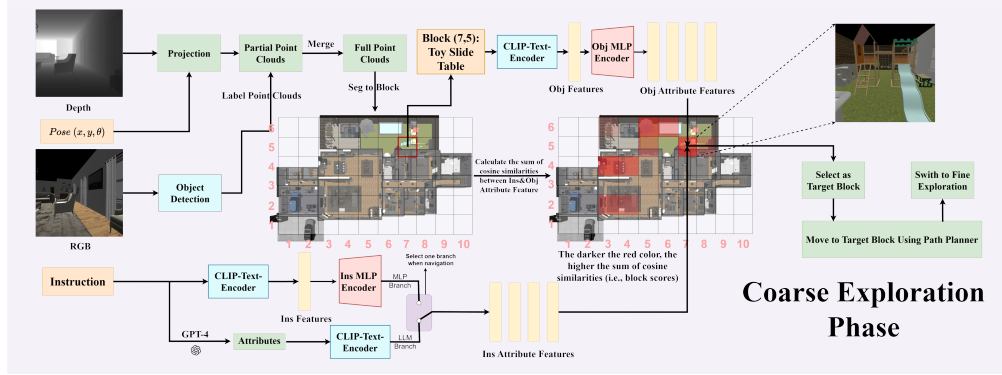
Figure 4: **Coarse Exploration.** This figure presents the process of building and labeling the point clouds, segmenting the blocks, and calculating the scores for each block.

4.2.1 Coarse Exploration Phase

In this phase (see Fig. 4), the agent receives the RGB-D input, pose, and the demand instruction. We then use the camera parameters and depth map to compute partial point clouds of the current observation and merge them with the previously observed point clouds. We use an object detection model, GLEE, to detect objects in the RGB image and project them into the depth map, labeling the point clouds. We segment the point clouds into many rectangular blocks according to x and y coordinates, and each block is a $b \times b$ square, where b is 2 in experiments. Thus, each detected object belongs to a block (according to the center of the object's point clouds). We use the instruction attribute features to query objects in each attribute block, and each block will then get a score. We use two different ways to generate instruction attribute features; see the LLM branch and MLP branch at the bottom in Fig. 4. **For the LLM branch** in Fig. 4, we use GPT-4 to generate language-level basic and preferred attributes separately and use CLIP-Text-Encoder to obtain basic and preferred attribute features. These two attribute features can be used to calculate two scores, basic and preferred scores. Adjusting the weights of these two scores can control whether to prioritize the search for basic or preferred solutions. The formula for calculating the score (*i.e.*, the process of query) is as follows:

$$ s = \sum_{o \in block} \left(r_p \times \max_{i=1..k_1;j=1..k_2} f^i_{pref\_ins} * f^j_o + r_b \times \max_{i=1..k_1;j=1..k_2} f^i_{basic\_ins} * f^j_o \right) \quad (3) $$

Where f^j_o denote j_{th} attribute features of the object o, $f^i_{basic\_ins}$ denote i_{th} basic attribute features of the instruction (so do $f^i_{pref\_ins}$), $*$ denotes cosine similarity, and r_b and r_p are adjustable weights for whether to find basic or preferred solutions. If deployed in a real environment, these two weights

7

can be freely adjusted by the user to apply to different situations. We conduct an ablation study on these two weights and discuss in detail how they affect the basic and preferred solutions in the experimental section. **For the MLP branch**, we use the Ins MLP Encoder from attribute training to map the instruction features into k_1 attribute features, and we use a similar query process to compute scores. MLP branch is a lightweight alternative that neither requires remote LLM nor consumes computational resources to run local LLMs. If deployed in a real environment, the agent is free to choose one of the branches according to the current LLM availability. We report the results of the two branches separately in the experimental section. Finally, we randomly choose a point in the highest-scoring and never-visited block as the waypoint. Then, the agent navigates to the point by a path-planning algorithm. Please see supplementary material for details about the coarse exploration module, the path-planning algorithm and blocks' score visualizations A.3.1.

4.2.2 Fine Exploration Phase
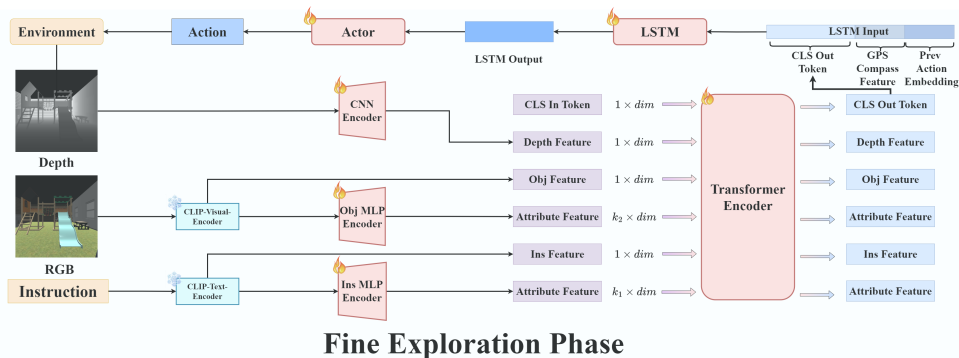
Figure 5: **Fine Exploration.** We employ imitation learning to train an end-to-end module in this phase. This module loads the Ins MLP Encoder and Obj MLP Encoder's parameters as initialization from attribute training, along with a Transformer Encoder to integrate features. The output feature corresponding to the CLS token is combined with GPS+Compass features and a previous action embedding and passed through an LSTM to generate actions by an actor.

In this phase, we train an end-to-end module using imitation learning similar to DDN. The alignment capabilities of CLIP in the visual and textual domains allow Obj MLP Encoder to still extract object attribute features through CLIP-Visual-Encoder's object features. The self-attention mechanism in the Transformer Encoder learns the association between the attribute features of the objects in the current field of view and the attribute features of the instructions, which can implicitly determine whether these objects need to be reported. Following BERT [103] and ViT [104], we add a CLS token as the output of the feature fusion. For more training and hyperparameter details, see the supplementary material A.3.2.

5 Experiment

5.1 Experimental Settings

We use habitat-sim and habitat-lab as our simulator and HSSD as our scene dataset. We randomly select 30 tasks as testing tasks and the remaining 270 tasks as training tasks(*i.e.*, unseen task and seen task in Tab. 5.3, respectively). These training tasks are used to collect trajectories to train the fine exploration module, VTN and ZSON. HSSD splits the scenes into val scenes and train scenes (*i.e.*, unseen scenes and seen scenes in Tab. 5.3, respectively). In all experimental settings, the judgment distance of the found list d_{find} is one meter, and a maximum number of Find n_{find} is five times, and the maximum step number n_{step} is 300. A single RTX 4090 is enough to run the experiments. More experimental settings are available in the supplementary material A.4.

8

Table 1: **Baseline Comparison.** Values in parentheses represent standard deviations. * represents the usage of ground truth semantic labels in the RGB image. The bold fonts represent optimal values.

| Method | Seen Scene | | | | | | | |
| | Seen Task | | | | Unseen Task | | | |
	SR_b	SR_p	SPL_b	SPL_p	SR_b	SR_p	SPL_b	SPL_p
Random	4.36 (0.36)	3.30 (0.07)	4.10 (0.45)	2.92 (0.04)	3.47 (0.27)	2.40 (0.22)	3.14 (0.26)	2.17 (0.25)
VTN	7.50 (3.01)	4.79 (3.02)	4.19 (1.16)	2.45 (1.51)	5.17 (1.53)	4.06 (1.15)	3.38 (0.59)	3.01 (0.99)
ZSON	6.75 (2.61)	3.79 (0.82)	4.27 (1.95)	2.41 (0.82)	4.33 (1.26)	3.08 (0.75)	3.02 (1.22)	2.18 (0.65)
DDN	8.10 (2.20)	6.62 (0.94)	5.95 (2.09)	4.73 (1.20)	6.70 (2.41)	4.52 (0.85)	4.67 (1.52)	3.14 (0.96)
MOPA+LLM*	18.61 (3.75)	13.25 (1.99)	3.00 (1.26)	2.18 (0.58)	14.67 (1.44)	**10.22 (1.28)**	2.03 (0.13)	1.55 (0.19)
FBE+LLM*	13.25 (0.35)	9.95 (2.3)	3.9 (0.71)	3.24 (0.96)	10.00 (1.80)	8.03 (2.23)	3.37 (0.54)	2.99 (0.73)
C2FAgent (MLP branch)	18.58 (2.78)	12.45 (2.30)	6.99 (1.72)	5.53 (1.05)	13.91 (2.91)	8.44 (0.87)	4.45 (0.77)	3.43 (0.41)
C2FAgent (LLM branch)	**23.82 (3.89)**	**14.28 (3.72)**	**7.94 (1.30)**	**5.80 (1.00)**	**15.93 (1.68)**	8.65 (0.71)	**6.21 (1.81)**	**4.60 (1.40)**

| Method | Unseen Scene | | | | | | | |
| | Seen Task | | | | Unseen Task | | | |
	SR_b	SR_p	SPL_b	SPL_p	SR_b	SR_p	SPL_b	SPL_p
Random	6.60 (0.67)	4.47 (0.47)	6.02 (0.71)	4.07 (0.44)	4.63 (0.52)	3.46 (0.40)	4.18 (0.50)	3.18 (0.42)
VTN	8.61 (4.25)	6.12 (3.84)	4.76 (1.96)	3.64 (1.86)	6.83 (1.04)	3.47 (1.55)	3.48 (0.15)	1.97 (0.56)
ZSON	6.89 (1.50)	5.06 (1.71)	4.02 (1.19)	3.46 (1.92)	5.33 (0.76)	2.94 (0.51)	2.70 (1.02)	1.91 (0.57)
DDN	10.00 (2.50)	5.93 (0.86)	7.89 (1.93)	4.86 (0.52)	7.10 (2.95)	4.70 (1.81)	5.37 (2.46)	3.48 (1.45)
MOPA+LLM*	17.83 (9.41)	14.17 (1.93)	3.06 (0.53)	2.96 (0.31)	10.33 (2.08)	6.72 (1.92)	1.19 (0.23)	0.78 (0.24)
FBE+LLM*	14.00 (4.47)	8.03 (2.23)	3.13 (0.66)	2.99 (0.73)	9.67 (1.53)	6.86 (1.04)	1.78 (0.58)	2.12 (0.68)
C2FAgent (MLP branch)	17.97 (2.87)	12.35 (2.49)	5.49 (1.23)	3.77 (0.70)	12.17 (3.71)	6.93 (1.63)	3.51 (1.05)	2.46 (0.59)
C2FAgent (LLM branch)	**23.06 (1.58)**	**15.98 (0.87)**	6.52 (0.91)	**5.11 (0.51)**	**15.88 (2.63)**	**9.48 (1.30)**	**6.03 (0.77)**	**4.50 (0.48)**

5.2 Baselines

Random is a method for randomly selecting actions. **VTN** [15] is an end-to-end closed-vocabulary ObjectGoal Navigation method. **ZSON** [21] is an end-to-end open-vocabulary ObjectGoal Navigation method. **MOPA+LLM** [43] is a modular-based Multi-object Navigation method using LLM to select target objects. **FBE+LLM** [59] is a modular-based Multi-object Navigation method that uses frontier-based exploration (FBE) and LLM to find objects. **DDN** [14] is an end-to-end Single-object Demand-driven Navigation method. For more details about baselines in training and method modifications, please see supplementary material A.4.2.

5.3 Baseline Comparison

The experimental results are in Tab. 5.3. Random reflects the difficulty of MO-DDN. The two end-to-end ON methods, VTN and ZSON, perform poorly, only slightly higher than Random. In comparison, though DDN uses attribute features designed for single object-instruction pairs, DDN's attribute features still provide some prior knowledge, outperforming VTN and ZSON. FBE+LLM and MOPA+LLM use different exploration strategies, but both use LLM to select waypoints. Both of them acquire the ground truth semantic label in the current RGB image when asking LLM whether or not to perform Find^5, and they outperform the three end-to-end methods. MOPA+LLM obtains suboptimal results in many cases, better than FBE+LLM, which may stem from the fact [43] that its exploration strategy is superior to FBE, allowing more objects to be detected.

Our method outperforms the baseline in the vast majority of settings. Attribute similarity scores are a good substitute for LLM for waypoint selection. Compared to MOPA+LLM, which requires 5.15 requests to the LLM per episode on average, C2FAgent (LLM branch) only needs to request the LLM to parse instructions into attributes once at the beginning and achieves better performance. Moreover, C2FAgent (MLP branch) does not need LLM parsing and gets results comparable to MOPA+LLM. We also find that when no object meets the demand in the current detected object list, each block has a low score, approximating a randomly selected block to explore. In contrast, when there exists an object that meets the demand, the corresponding block has a significantly higher score than the other blocks. This indicates that the attribute model can balance exploration and exploitation on its own.

5.4 Ablation Study

In this section, we would like to discuss the following four questions:

- **Q1**: Is selecting waypoints by attribute feature similarity scores better than FBE, LLM and CLIP features' similarity scores?

---

[5]when other actions are performed, the ground truth semantic label will not be provided, and only the GLEE [31] detection will be provided.

9

Table 2: Ablation on Coarse Exploration (Q1)

Method	SR_b	SR_p	SPL_b	SPL_p
Coarse+Fine (Ours)	**23.82 (3.89)**	**14.28 (3.72)**	**7.94 (1.30)**	**5.80 (1.00)**
FBE+Fine	14.56 (5.57)	11.36 (5.45)	4.94 (0.84)	3.92 (0.95)
LLM+Fine	13.69 (4.91)	9.47 (2.57)	5.12 (0.72)	3.32 (0.07)
CLIP+Fine	12.51 (3.85)	7.65 (2.77)	4.52 (0.87)	3.05 (0.43)

Table 3: Ablation on Fine Exploration (Q2)

Method	SR_b	SR_p	SPL_b	SPL_p
Coarse+Fine (Ours)	**23.82 (3.89)**	**14.28 (3.72)**	**7.94 (1.30)**	**5.80 (1.00)**
Coarse+ZSON	8.74 (5.07)	4.21 (2.0)	6.72 (4.29)	3.45 (2.0)
Coarse+VTN	16.89 (4.57)	10.87 (4.73)	5.36 (1.11)	3.83 (1.32)
Coarse+Random	5.82 (1.22)	4.63 (0.93)	4.04 (2.52)	3.27 (2.12)

Table 4: Ablation on Attribute Training (Q3)

Method	SR_b	SR_p	SPL_b	SPL_p
Ours	**23.82 (3.89)**	**14.28 (3.72)**	**7.94 (1.30)**	**5.80 (1.00)**
Ours w/o VQ-VAE	18.34 (1.33)	11.42 (0.23)	5.63 (0.72)	4.72 (0.63)
Ours w/o codebook init	17.36 (2.32)	12.02 (2.54)	5.73 (1.14)	4.56 (1.18)

Table 5: Ablation on Score Weights (Q4)

Method (C2FAgent)	SR_b	SR_p	SPL_b	SPL_p
$r_b = 1, r_p = 2$	19.89 (2.00)	**15.05 (1.58)**	7.91 (1.32)	**6.20 (0.64)**
$r_b = 1, r_p = 1$	23.82 (3.89)	14.28 (3.72)	7.94 (1.30)	5.80 (1.00)
$r_b = 1, r_p = 0$	**25.10 (2.06)**	12.43 (1.04)	**9.26 (2.07)**	5.34 (1.09)

- **Q2**: Do attribute features also work in the end-to-end fine exploration modules? How about replacing the fine exploration module with VTN and ZSON?

- **Q3**: Do VQ-VAE losses and codebook initialization contribute to experimental results?

- **Q4**: Can adjusting the weights of basic and preferred scores affect agent behavior?

In the ablation study, we report the results in the seen tasks and seen scenes. Ours refers to C2FAgent (LLM branch). The experimental results for all four questions are in Tab. 5.4. For specific experimental setups, please see supplementary materials A.4.3.

For Q1, the experimental results demonstrate that attribute-based coarse exploration outperforms rule-based FBE, commonsense-based LLM and CLIP-based exploration. **For Q2**, the experimental results show that the fine exploration module utilizes the prior in the attribute model well, outperforming the VTN, which has larger model parameters, and the ZSON, which has been pre-trained on 36M total episodes and fine-tuned on the same trajectory dataset with Ours. In addition, we note that Coarse+VTN exceeds VTN, suggesting that the coarse exploration module can steer the agent to the region where it is more likely to find objects that satisfy the demand. **For Q3**, we find that the performance decreases after removing the VQ-VAE Loss or codebook initialization. Since the attribute model itself is trained on language-grounding tasks and thus agnostic to the task being evaluated, we can argue that the VQ-VAE Loss and the initialization contribute to the generalizability of attribute features. **For Q4**, adjusting the weights of two scores can indeed affect the agent's behavior. By tuning up r_p, SR_p and SPL_p increase, while SR_b and SPL_b decrease; and vice versa. This characteristic allows the user to freely decide whether to prioritize the search for the basic or preferred solution in the current situation. For example, when a user who likes Coke is **very** thirsty, he can increase r_b to allow the agent to satisfy the basic demand with a higher success rate, while in the general case, r_p can be increased to allow the agent to try to satisfy the preferred demand.

6 Conclusion and Discussion

In this paper, we propose a new benchmark, MO-DDN, which can be regarded as a multi-object version of DDN. Moreover, MO-DDN can be considered a crucial preliminary step in task planning. We also propose a coarse-to-fine attribute-based exploration agent, extending the concept of "attribute" to a multi-object setting. The agent uses attribute features in different exploration phases. The experimental results show that our method outperforms the baselines, and the attribute features improve exploration efficiency. The ablation study also demonstrated the effectiveness of our method.

Limitations and Broader Societal Impacts In this paper, we assume that the number of attributes of both instructions and objects is fixed values (*i.e.*, k_1 and k_2). Though this facilitates training and its experimental results outperform baselines, there are still some gaps with real life. Future work could consider more flexible attribute features. Please see A.5 for more limitation discussion. To the best of our knowledge, there are no observable adverse effects on society.

Acknowledgments and Disclosure of Funding

This work was supported by The National Youth Talent Support Program (8200800081), National Natural Science Foundation of China (No. 62376006) and National Natural Science Foundation of China (No. 62136001).

10

References

[1] R. J. Taormina and J. H. Gao, "Maslow and the motivation hierarchy: Measuring satisfaction of the needs," *The American journal of psychology*, vol. 126, no. 2, pp. 155–177, 2013.

[2] J. E. Gawel, "Herzberg's theory of motivation and maslow's hierarchy of needs," *Practical Assessment, Research, and Evaluation*, vol. 5, no. 1, p. 11, 2019.

[3] A. Maslow, *Motivation And Personality: Motivation And Personality: Unlocking Your Inner Drive and Understanding Human Behavior by A. H. Maslow*.   Prabhat Prakashan, 1981. [Online]. Available: https://books.google.com/books?id=DVmxDwAAQBAJ

[4] A. H. Maslow, "A dynamic theory of human motivation." 1958.

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

[6] J. Liang, W. Huang, F. Xia, P. Xu, K. Hausman, B. Ichter, P. Florence, and A. Zeng, "Code as policies: Language model programs for embodied control," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*.   IEEE, 2023, pp. 9493–9500.

[7] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar *et al.*, "Inner monologue: Embodied reasoning through planning with language models," *arXiv preprint arXiv:2207.05608*, 2022.

[8] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, J. Dabis, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, J. Hsu *et al.*, "Rt-1: Robotics transformer for real-world control at scale," *arXiv preprint arXiv:2212.06817*, 2022.

[9] D. Driess, F. Xia, M. S. Sajjadi, C. Lynch, A. Chowdhery, B. Ichter, A. Wahid, J. Tompson, Q. Vuong, T. Yu *et al.*, "Palm-e: An embodied multimodal language model," *arXiv preprint arXiv:2303.03378*, 2023.

[10] A. Brohan, N. Brown, J. Carbajal, Y. Chebotar, X. Chen, K. Choromanski, T. Ding, D. Driess, A. Dubey, C. Finn *et al.*, "Rt-2: Vision-language-action models transfer web knowledge to robotic control," *arXiv preprint arXiv:2307.15818*, 2023.

[11] W. Huang, C. Wang, R. Zhang, Y. Li, J. Wu, and L. Fei-Fei, "Voxposer: Composable 3d value maps for robotic manipulation with language models," *arXiv preprint arXiv:2307.05973*, 2023.

[12] R. Wu and Y. Zhao, "Vat-mart: Learning visual action trajectory proposals for manipulating 3d articulated objects," in *International Conference on Learning Representations (ICLR), 2022*, 2022.

[13] G. Zhou, Y. Hong, and Q. Wu, "Navgpt: Explicit reasoning in vision-and-language navigation with large language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 7, 2024, pp. 7641–7649.

[14] H. Wang, A. G. H. Chen, X. Li, M. Wu, and H. Dong, "Find what you want: Learning demand-conditioned object attribute space for demand-driven navigation," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[15] H. Du, X. Yu, and L. Zheng, "Vtnet: Visual transformer network for object goal navigation," *arXiv preprint arXiv:2105.09447*, 2021.

[16] W. Cai, S. Huang, G. Cheng, Y. Long, P. Gao, C. Sun, and H. Dong, "Bridging zero-shot object navigation and foundation models through pixel-guided navigation skill," *arXiv preprint arXiv:2309.10309*, 2023.

[17] D. S. Chaplot, D. P. Gandhi, A. Gupta, and R. R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," *Advances in Neural Information Processing Systems*, vol. 33, pp. 4247–4258, 2020.

11

[18] F. Zhu, X. Liang, Y. Zhu, Q. Yu, X. Chang, and X. Liang, "Soon: Scenario oriented object navigation with graph-based exploration," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 689–12 699.

[19] R. Fukushima, K. Ota, A. Kanezaki, Y. Sasaki, and Y. Yoshiyasu, "Object memory transformer for object goal navigation," in *2022 International Conference on Robotics and Automation (ICRA)*. IEEE, 2022, pp. 11 288–11 294.

[20] K. Zhou, K. Zheng, C. Pryor, Y. Shen, H. Jin, L. Getoor, and X. E. Wang, "Esc: Exploration with soft commonsense constraints for zero-shot object navigation," in *International Conference on Machine Learning*. PMLR, 2023, pp. 42 829–42 842.

[21] A. Majumdar, G. Aggarwal, B. Devnani, J. Hoffman, and D. Batra, "Zson: Zero-shot object-goal navigation using multimodal goal embeddings," *Advances in Neural Information Processing Systems*, vol. 35, pp. 32 340–32 352, 2022.

[22] C. Li, R. Zhang, J. Wong, C. Gokmen, S. Srivastava, R. Martín-Martín, C. Wang, G. Levine, M. Lingelbach, J. Sun *et al.*, "Behavior-1k: A benchmark for embodied ai with 1,000 everyday activities and realistic simulation," in *Conference on Robot Learning*. PMLR, 2023, pp. 80–93.

[23] P. Marza, L. Matignon, O. Simonin, and C. Wolf, "Multi-object navigation with dynamically learned neural implicit representations," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 11 004–11 015.

[24] S. Wani, S. Patel, U. Jain, A. Chang, and M. Savva, "Multion: Benchmarking semantic map memory using multi-object navigation," *Advances in Neural Information Processing Systems*, vol. 33, pp. 9700–9712, 2020.

[25] P. Chen, D. Ji, K. Lin, W. Hu, W. Huang, T. Li, M. Tan, and C. Gan, "Learning active camera for multi-object navigation," *Advances in Neural Information Processing Systems*, vol. 35, pp. 28 670–28 682, 2022.

[26] X. Liu, H. Palacios, and C. Muise, "Egocentric planning for scalable embodied task achievement," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[27] B. Y. Lin, C. Huang, Q. Liu, W. Gu, S. Sommerer, and X. Ren, "On grounded planning for embodied tasks with language models," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 37, no. 11, 2023, pp. 13 192–13 200.

[28] Z. Wu, Z. Wang, X. Xu, J. Lu, and H. Yan, "Embodied task planning with large language models," *arXiv preprint arXiv:2307.01848*, 2023.

[29] J. Mendez-Mendez, L. P. Kaelbling, and T. Lozano-Pérez, "Embodied lifelong learning for task and motion planning," in *Conference on Robot Learning*. PMLR, 2023, pp. 2134–2150.

[30] T. Cheng, L. Song, Y. Ge, W. Liu, X. Wang, and Y. Shan, "Yolo-world: Real-time open-vocabulary object detection," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2024.

[31] J. Wu, Y. Jiang, Q. Liu, Z. Yuan, X. Bai, and S. Bai, "General object foundation model for images and videos at scale," 2024.

[32] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds *et al.*, "Flamingo: a visual language model for few-shot learning," *Advances in neural information processing systems*, vol. 35, pp. 23 716–23 736, 2022.

[33] Z. Yang, L. Li, K. Lin, J. Wang, C.-C. Lin, Z. Liu, and L. Wang, "The dawn of lmms: Preliminary explorations with gpt-4v (ision)," *arXiv preprint arXiv:2309.17421*, vol. 9, no. 1, p. 1, 2023.

[34] J. Chen, D. Zhu, X. Shen, X. Li, Z. Liu, P. Zhang, R. Krishnamoorthi, V. Chandra, Y. Xiong, and M. Elhoseiny, "Minigpt-v2: large language model as a unified interface for vision-language multi-task learning," *arXiv preprint arXiv:2310.09478*, 2023.

12

[35] D. Zhu, J. Chen, X. Shen, X. Li, and M. Elhoseiny, "Minigpt-4: Enhancing vision-language understanding with advanced large language models," *arXiv preprint arXiv:2304.10592*, 2023.

[36] W. Wang, Z. Chen, X. Chen, J. Wu, X. Zhu, G. Zeng, P. Luo, T. Lu, J. Zhou, Y. Qiao *et al.*, "Visionllm: Large language model is also an open-ended decoder for vision-centric tasks," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[37] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[38] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann *et al.*, "Palm: Scaling language modeling with pathways," *Journal of Machine Learning Research*, vol. 24, no. 240, pp. 1–113, 2023.

[39] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[40] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale *et al.*, "Llama 2: Open foundation and fine-tuned chat models," *arXiv preprint arXiv:2307.09288*, 2023.

[41] Y. Shen, K. Song, X. Tan, D. Li, W. Lu, and Y. Zhuang, "Hugginggpt: Solving ai tasks with chatgpt and its friends in hugging face," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[42] G. Team, R. Anil, S. Borgeaud, Y. Wu, J.-B. Alayrac, J. Yu, R. Soricut, J. Schalkwyk, A. M. Dai, A. Hauth *et al.*, "Gemini: a family of highly capable multimodal models," *arXiv preprint arXiv:2312.11805*, 2023.

[43] S. Raychaudhuri, T. Campari, U. Jain, M. Savva, and A. X. Chang, "Mopa: Modular object navigation with pointgoal agents," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 5763–5773.

[44] S. Gupta, J. Davidson, S. Levine, R. Sukthankar, and J. Malik, "Cognitive mapping and planning for visual navigation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 2616–2625.

[45] H. Luo, A. Yue, Z.-W. Hong, and P. Agrawal, "Stubborn: A strong baseline for indoor object navigation," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2022, pp. 3287–3293.

[46] S. Rudra, S. Goel, A. Santara, C. Gentile, L. Perron, F. Xia, V. Sindhwani, C. Parada, and G. Aggarwal, "A contextual bandit approach for learning to plan in environments with probabilistic goal configurations," in *2023 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2023, pp. 5645–5652.

[47] Y. Liang, B. Chen, and S. Song, "Sscnav: Confidence-aware semantic scene completion for visual semantic navigation," in *2021 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2021, pp. 13 194–13 200.

[48] G. Georgakis, B. Bucher, K. Schmeckpeper, S. Singh, and K. Daniilidis, "Learning to map for active semantic goal navigation," *arXiv preprint arXiv:2106.15648*, 2021.

[49] S. Y. Min, Y.-H. H. Tsai, W. Ding, A. Farhadi, R. Salakhutdinov, Y. Bisk, and J. Zhang, "Self-supervised object goal navigation with in-situ finetuning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2023, pp. 7119–7126.

[50] G. Kumar, N. S. Shankar, H. Didwania, R. D. Roychoudhury, B. Bhowmick, and K. M. Krishna, "Gcexp: Goal-conditioned exploration for object goal navigation," in *2021 30th IEEE International Conference on Robot & Human Interactive Communication (RO-MAN)*. IEEE, 2021, pp. 123–130.

13

[51] W. B. Shen, D. Xu, Y. Zhu, L. J. Guibas, L. Fei-Fei, and S. Savarese, "Situational fusion of visual representation for visual navigation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2019, pp. 2881–2890.

[52] A. Mousavian, A. Toshev, M. Fišer, J. Košecká, A. Wahid, and J. Davidson, "Visual representations for semantic target driven navigation," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8846–8852.

[53] R. Druon, Y. Yoshiyasu, A. Kanezaki, and A. Watt, "Visual object search by learning spatial context," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 1279–1286, 2020.

[54] W. Yang, X. Wang, A. Farhadi, A. Gupta, and R. Mottaghi, "Visual semantic navigation using scene priors," *arXiv preprint arXiv:1810.06543*, 2018.

[55] A. Pal, Y. Qiu, and H. Christensen, "Learning hierarchical relationships for object-goal navigation," in *Conference on Robot Learning*. PMLR, 2021, pp. 517–528.

[56] B. Mayo, T. Hazan, and A. Tal, "Visual navigation with spatial attention," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2021, pp. 16 898–16 907.

[57] R. Dang, Z. Shi, L. Wang, Z. He, C. Liu, and Q. Chen, "Unbiased directed object attention graph for object navigation," in *Proceedings of the 30th ACM International Conference on Multimedia*, 2022, pp. 3617–3627.

[58] J. Ye, D. Batra, A. Das, and E. Wijmans, "Auxiliary tasks and exploration enable objectgoal navigation," in *Proceedings of the IEEE/CVF international conference on computer vision*, 2021, pp. 16 117–16 126.

[59] B. Yamauchi, "A frontier-based approach for autonomous exploration," in *Proceedings 1997 IEEE International Symposium on Computational Intelligence in Robotics and Automation CIRA'97.'Towards New Computational Principles for Robotics and Automation'*. IEEE, 1997, pp. 146–151.

[60] P. Anderson, A. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, V. Koltun, J. Kosecka, J. Malik, R. Mottaghi, M. Savva *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[61] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. Van Den Hengel, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2018, pp. 3674–3683.

[62] F. Zhu, Y. Zhu, X. Chang, and X. Liang, "Vision-language navigation with self-supervised auxiliary reasoning tasks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 012–10 022.

[63] H. Wang, W. Wang, W. Liang, C. Xiong, and J. Shen, "Structured scene memory for vision-language navigation," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2021, pp. 8455–8464.

[64] A. Majumdar, A. Shrivastava, S. Lee, P. Anderson, D. Parikh, and D. Batra, "Improving vision-and-language navigation with image-text pairs from the web," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*. Springer, 2020, pp. 259–274.

[65] X. Wang, Q. Huang, A. Celikyilmaz, J. Gao, D. Shen, Y.-F. Wang, W. Y. Wang, and L. Zhang, "Vision-language navigation policy learning and adaptation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 43, no. 12, pp. 4205–4216, 2020.

[66] Y. Long, X. Li, W. Cai, and H. Dong, "Discuss before moving: Visual language navigation via multi-expert discussions," *arXiv preprint arXiv:2309.11382*, 2023.

[67] Y. Hong, Q. Wu, Y. Qi, C. Rodriguez-Opazo, and S. Gould, "Vln bert: A recurrent vision-and-language bert for navigation," in *Proceedings of the IEEE/CVF conference on Computer Vision and Pattern Recognition*, 2021, pp. 1643–1653.

14

[68] D. Shah, B. Osiński, S. Levine *et al.*, "Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action," in *Conference on robot learning*. PMLR, 2023, pp. 492–504.

[69] C. Chen, Z. Al-Halah, and K. Grauman, "Semantic audio-visual navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 15 516–15 525.

[70] C. Chen, U. Jain, C. Schissler, S. V. A. Gari, Z. Al-Halah, V. K. Ithapu, P. Robinson, and K. Grauman, "Soundspaces: Audio-visual navigation in 3d environments," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part VI 16*. Springer, 2020, pp. 17–36.

[71] C. Chen, S. Majumder, Z. Al-Halah, R. Gao, S. K. Ramakrishnan, and K. Grauman, "Learning to set waypoints for audio-visual navigation," *arXiv preprint arXiv:2008.09622*, 2020.

[72] H. Wang, Y. Wang, F. Zhong, M. Wu, J. Zhang, Y. Wang, and H. Dong, "Learning semantic-agnostic and spatial-aware representation for generalizable visual-audio navigation," *IEEE Robotics and Automation Letters*, 2023.

[73] Y. Yu, W. Huang, F. Sun, C. Chen, Y. Wang, and X. Liu, "Sound adversarial audio-visual navigation," *arXiv preprint arXiv:2202.10910*, 2022.

[74] A. Younes, D. Honerkamp, T. Welschehold, and A. Valada, "Catch me if you hear me: Audio-visual navigation in complex unmapped environments with moving sounds," *IEEE Robotics and Automation Letters*, vol. 8, no. 2, pp. 928–935, 2023.

[75] M. Hussain, "Yolo-v1 to yolo-v8, the rise of yolo and its complementary nature toward digital manufacturing and industrial defect detection," *Machines*, vol. 11, no. 7, p. 677, 2023.

[76] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," in *European conference on computer vision*. Springer, 2020, pp. 213–229.

[77] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo *et al.*, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4015–4026.

[78] L. Ke, M. Ye, M. Danelljan, Y.-W. Tai, C.-K. Tang, F. Yu *et al.*, "Segment anything in high quality," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[79] S. Ren, F. Luzi, S. Lahrichi, K. Kassaw, L. M. Collins, K. Bradbury, and J. M. Malof, "Segment anything, from space?" in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2024, pp. 8355–8365.

[80] P. O. O Pinheiro, R. Collobert, and P. Dollár, "Learning to segment object candidates," *Advances in neural information processing systems*, vol. 28, 2015.

[81] J. Kim, E. S. Lee, M. Lee, D. Zhang, and Y. M. Kim, "Sgolam: Simultaneous goal localization and mapping for multi-object goal navigation," *arXiv preprint arXiv:2110.07171*, 2021.

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

[83] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar *et al.*, "Llama: Open and efficient foundation language models," *arXiv preprint arXiv:2302.13971*, 2023.

[84] G. Team, T. Mesnard, C. Hardin, R. Dadashi, S. Bhupatiraju, S. Pathak, L. Sifre, M. Rivière, M. S. Kale, J. Love *et al.*, "Gemma: Open models based on gemini research and technology," *arXiv preprint arXiv:2403.08295*, 2024.

[85] C. H. Song, J. Wu, C. Washington, B. M. Sadler, W.-L. Chao, and Y. Su, "Llm-planner: Few-shot grounded planning for embodied agents with large language models," 2023.

15

[86] Z. Zhao, W. S. Lee, and D. Hsu, "Large language models as commonsense knowledge for large-scale task planning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[87] Z. Shao, Z. Yu, M. Wang, and J. Yu, "Prompting large language models with answer heuristics for knowledge-based visual question answering," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 14 974–14 983.

[88] S. Yu, J. Cho, P. Yadav, and M. Bansal, "Self-chained image-language model for video localization and question answering," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[89] J. Guo, J. Li, D. Li, A. M. H. Tiong, B. Li, D. Tao, and S. Hoi, "From images to textual prompts: Zero-shot visual question answering with frozen large language models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 10 867–10 877.

[90] J. Cui, Z. Li, Y. Yan, B. Chen, and L. Yuan, "Chatlaw: Open-source legal large language model with integrated external knowledge bases," *arXiv preprint arXiv:2306.16092*, 2023.

[91] Y. Zhuang, Y. Yu, K. Wang, H. Sun, and C. Zhang, "Toolqa: A dataset for llm question answering with external tools," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[92] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*. PMLR, 2021, pp. 8748–8763.

[93] V. S. Dorbala, G. Sigurdsson, R. Piramuthu, J. Thomason, and G. S. Sukhatme, "Clip-nav: Using clip for zero-shot vision-and-language navigation," *arXiv preprint arXiv:2211.16649*, 2022.

[94] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi, "Simple but effective: Clip embeddings for embodied ai," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 14 829–14 838.

[95] P. Kinghorn, L. Zhang, and L. Shao, "A region-based image caption generator with refined descriptions," *Neurocomputing*, vol. 272, pp. 416–424, 2018.

[96] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan, "Show and tell: A neural image caption generator," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 3156–3164.

[97] X. Chen and C. Lawrence Zitnick, "Mind's eye: A recurrent visual representation for image caption generation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 2422–2431.

[98] T. Wang, J. Zhang, J. Fei, Y. Ge, H. Zheng, Y. Tang, Z. Li, M. Gao, S. Zhao, Y. Shan *et al.*, "Caption anything: Interactive image description with diverse multimodal controls," *arXiv preprint arXiv:2305.02677*, 2023.

[99] M. Khanna*, Y. Mao*, H. Jiang, S. Haresh, B. Shacklett, D. Batra, A. Clegg, E. Undersander, A. X. Chang, and M. Savva, "Habitat Synthetic Scenes Dataset (HSSD-200): An Analysis of 3D Scene Scale and Realism Tradeoffs for ObjectGoal Navigation," *arXiv preprint*, 2023.

[100] S. Lloyd, "Least squares quantization in pcm," *IEEE transactions on information theory*, vol. 28, no. 2, pp. 129–137, 1982.

[101] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *Advances in neural information processing systems*, vol. 30, 2017.

[102] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song, "Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 23 171–23 181.

[103] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

16

[104] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, J. Uszkoreit, and N. Houlsby, "An image is worth 16x16 words: Transformers for image recognition at scale," *ICLR*, 2021.

17

A Appendix / supplemental material

- Multi-object Demand-driven Navigation Task A.1
    - Task Dataset Generation A.1.1
    - Task Metrics A.1.2
    - Task Dataset Statistics A.1.3
- Attribute Feature Training A.2
    - Training Data Preparation A.2.1
    - Attribute Training Details A.2.2
- Details about Coarse-to-Fine Exploration A.3
    - Design Details and Visualizations for Coarse Exploration Module A.3.1
    - Training Details about Fine Exploration Module A.3.2
- Experiments A.4
    - Details about Experimental Settings A.4.1
    - Details about Baselines A.4.2
    - Details about Ablation Study A.4.3
    - Details about LLM's Prompt in Experiments A.4.4
- More Limitations A.5

A.1 Multi-object Demand-driven Navigation Task

A.1.1 Task Dataset Generation

We base our instruction generation process on the 466 object categories in the HSSD dataset [99]. In short, we perform three steps while generating the task: LLM pre-generation, LLM revision, and manual review (we use gpt-4-0125-preview API in this paper). In the first step, we prompt the LLM with task definition, available object category list, task format, and generation guidelines to generate the raw tasks. In the second step, we prompt the LLM with previous raw tasks one by one. We let the LLM check whether the input raw task is in the task format, whether the objects involved are in the given object category list, and whether the demand instructions and the solutions correspond to each other, and let the LLM state the reason for the revision before providing the result of the revision. In the third step, we manually check the generated tasks, removing overly far-fetched tasks and adding more obvious solutions.

Here are some examples. For each example, we only show part of the solution.

task instruction: I need to display my photography collection, preferably with good lighting.
basic solution: [picture frame, bookshelf]
preferred solution: [picture frame, bookshelf, table lamp], [picture frame, bookshelf, ceiling lamp]

task instruction: I need a sleeping arrangement for a guest staying over one night; I hope the bed is comfortable enough.
basic solution: [single bed, blanket, pillow], [sofa, blanket]
preferred solution: [king bed, blanket, pillow], [double bed, blanket, pillow]

task instruction: Train my small dog with water and treats, preferably it is for pet only.
basic solution: [mixing bowl], [pet bowl], [bowl]
preferred solution: [pet bowl]

task instruction: I need to work on a writing project but prefer a quiet and comfortable space.
basic solution: [desk, swivel chair, notebook], [desk, straight chair, laptop]
preferred solution: [desk, swivel chair, notebook, earphone, room divider]

We use the following prompts to generate the raw tasks:

System Prompt: You are an AI assistant that can understand human demands and imagine what human demands can be met with existing object categories.

18

Prompt:
Task Generation: Demand-Driven Navigation
Objective: Create a navigation task where an agent must locate an object within a specific category that satisfies given demands and preferences.
Category:
bed, sofa, cup, desktop....
Task Requirements:
- **Basic Demand:** Describe the fundamental requirement for the object.
- **Preference:** Detail any additional preferences that refine the object selection.

###Demand-driven Navigation Task Template###
{
"task_instruction": $basic_demand$, $preference$
"basic_demand_instruction": "String"
"preferred_demand_instruction": "String"
"basic_solution": [[object_a, object_c], [object_b]]
"preferred_solution": [[object_a, object_c, object_d], [object_b, object_f]]
}
This should be in dict format.
basic_solutio is a list whose elements are lists, and each element represents a solution that meets the basic_demand_instruction; each solution may consist of one or more objects.
preferred_solution is a list whose elements are lists, and each element represents a solution that meets both the basic_demand_instruction and preference; each solution may consist of one or more objects.
"[object_g]" represents that just object_g can meet the demand.
"[object_x, object_y, object_z]" represents that only the combination of object x, y, and z can meet the demand.

Example
$Example Tasks$

###Additional Guideline ###
Avoid demands related to needing a specific place or location.
Focus on generating novel demands, possibly in the realm of entertainment, that don't require a specific place.
Finalize the task using the provided template, ensuring it is concise and formatted correctly.
Whenever possible, generate task_instruction that requires multiple combinations of objects to be met, such as [object_a, object_c, object_d] and [object_b, object_c, object_e].

Previously Generated Task Instructions:
$Previous Task Instruction Example$

Process
Determine "basic_demand_instruction" and "preferred_demand_instruction" based on the object category.
Sequentially consider how each object or combination thereof meets the demands.
Finalize your response in the provided dict format, ensuring logical consistency between "basic_solution" and "preferred_solution."
Make the response more concise and clear and can be executed as "json.loads(task)".
Please sequentially generate ten tasks split by "========", and make sure the task is totally different from the previously generated task instructions.

We use the following prompts to revise the raw tasks:

System Prompt: You are an AI assistant that can help check whether objects meet the demands, equipped with common life knowledge. Your reply should be in JSON string format.

19

Prompt:
Task Verification and Modification
Objective: Evaluate the provided task for logical consistency and feasibility. Modify it as necessary to ensure all requirements are met accurately.
###Object Category###
bed, sofa, cup, desktop....

###Demand-driven Navigation Task Template###
{
"task_instruction": $basic_demand$, $preference$
"basic_demand_instruction": "String"
"preferred_demand_instruction": "String"
"basic_solution": [[object_a, object_c], [object_b]]
"preferred_solution": [[object_a, object_c, object_d], [object_b, object_f]]
}
This should be in dict format.
basic_solution is a list whose elements are lists, and each element represents a solution that meets the basic_demand_instruction; each solution may consist of one or more objects.
preferred_solution is a list whose elements are lists, and each element represents a solution that meets both the basic_demand_instruction and preference; each solution may consist of one or more objects.
"[object_g]" represents that just object_g can meet the demand.
"[object_x, object_y, object_z]" represents that only the combination of object x, y, and z can meet the demand.

###To be modified Task in JSON String Format###
$Task Dict$

Instructions:
1. Verify Object Categories: Ensure all items in 'basic_solution' and 'preferred_solution' are from the provided Object Category. Replace or remove any items that do not belong.
2. Expand Object Lists: Identify additional items within the Object Category that can fulfill the 'basic_demand_instruction' and 'preferred_demand_instruction'. Add these items to 'basic_solution' and 'preferred_solution' as appropriate.
3. Validate Combination Solutions: Assess if each combination of objects in 'basic_solution' meets the 'basic_demand_instruction' and if removing any object from these combinations makes them invalid. Apply the same verification for 'preferred_solution' concerning both 'basic_demand_instruction' and 'preferred_demand_instruction'.
4. Ensure Subset Relationship: Confirm 'preferred_solution' is a subset of 'basic_solution'. If not, integrate missing combinations from 'preferred_solution' into 'basic_solution'.

Evaluation and Modification Steps:
- Begin by reviewing the JSON string of the current task for any logical inconsistencies or missing elements.
- Identify and explain any aspects that are not rational or feasible within the context of the given Object Category.
- Propose modifications, ensuring all suggested items are included in the Object Category and adhere to the Demand-driven Navigation Task Template.
Final Instruction:
- Present a brief critique of the task's initial setup, highlighting any irrational elements. - Offer a detailed plan for rectification, including specific changes to 'basic_solution' and 'preferred_solution'. Ensure these modifications respect the original Object Category and meet the template requirements.
- Your response should conclude with a revised JSON string format of the task, reflecting all necessary adjustments for coherence and completeness.

20

Table 6: A Comparison Between MO-DDN and DDN. The slash symbol "/" distinguishes between basic and preferred data. The left side of the slash indicates basic, and the right side indicates preferred.

	MO-DDN	DDN
Preference	✓	
Multi-object	✓	
Number of Object Category	358	109
Average Instruction Length	17.44	7.5
Average Number of Solution	17.51/50.4	2.3
Average Number of Object Per Solution	2.41/3.59	1

A.1.2 Task Metrics

$$SR_{basic} = \frac{1}{N} \sum_{i=1}^{N} \max_{s_b \in So_b} \frac{\sum_{o \in FL} \mathbb{1}_{o \in s_b}}{Len(s_b)} \qquad (4)$$

An example of calculating SR_{basic} in an episode: when the agent finds object [a, b, c, d, e, f] and the solutions are object [a, b, c, x, y] or [d, e, m, n], the SR_{basic} is $max(\frac{len(a,b,c)}{len(a,b,c,x,y)}, \frac{len(d,e)}{len(d,e,m,n)})$, *i.e.*, 0.6.

We calculate the SPL [60] using the following formula:

$$SPL_{basic} = \frac{1}{N} \sum_{i=1}^{N} SR_{basic}^{i} \frac{l_i}{max(p_i, l_i)} \qquad (5)$$

where SR_{basic}^{i} is the success rate of i_{th} episode, *i.e.*, $SR_{basic}^{i} = \max_{s_b \in So_b} \frac{\sum_{o \in FL} \mathbb{1}_{o \in s_b}}{Len(s_b)}$, l_i is the length of the shortest path of an episode, and p_i is the length of the path taken by the agent in an episode.

A.1.3 Task Dataset Statistics

In MO-DDN, we generate 300 tasks, encompassing 358 object categories from the HSSD dataset. Notably, the average instruction length in MO-DDN is 17.44, significantly longer than the average length of 7.5 in DDN. On average, there are 17.51 basic solutions and 50.4 preferred solutions per task (note that preferred solutions also include basic demands so that the number will be higher than just basic solutions), much more than the average solution number of 2.3 in DDN. Each basic solution will contain 2.41 objects, and each preferred solution will contain 3.59 objects, while only one object is in the solution of DDN. In general, MO-DDN increases the complexity of instructions and the diversity of solutions. This makes MO-DDN more relevant to real-life environments and more difficult. We create a table to compare the differences between DDN and MO-DDN shown in Tab. 6.

A.2 Attribute Feature Training

A.2.1 Training Data Preparation

Due to the restricted scope of object categories within the HSSD dataset, the tasks are limited to 358 object categories, a significantly smaller subset than those observed in real-world scenarios, which may cause overfitting on trained task and object categories. To address this limitation and enhance attribute feature generalization (*i.e.*, we would like the attribute feature to be universal and not limited to the HSSD dataset), we introduce language-grounding tasks by prompting GPT-4 to generate tasks without constraints on object categories and learn attribute features over these language-grounding tasks. In these tasks, a broader array of objects and instructions is employed, thereby fostering improved diversity and generalization of attribute features. Then, for each language-grounding task, we ask GPT-4 to answer the questions "What attributes are required to fulfill these demands?" and "What attributes are inherent to these objects?" This process yields language-level attributes for instructions and objects, which is used for training attribute features.

21

We use the following prompts to generate language-grounding tasks:

System Prompt: You are a household task generator. Generate a list of ten daily household tasks or needs that a person might have. These should be tasks that are commonly encountered in a daily living environment.

Prompt:
Task Introduction
Generate a list of ten daily household tasks or needs that a person might have. These should be tasks that are commonly encountered in a daily living environment.

Task Example
$Some Task Examples$

Task Format
String: List[List[String]]
For example, task_instruction: [["ObjectA", "ObjectB"], ["ObjectC", "ObjectD", "ObjectE"], ["ObjectF"], ["ObjectG", "ObjectH"]]

Instructions
- Generate a list of 10 unique daily household tasks or needs and separate them with "=========".
- Generated ten tasks should be distinct from each other and Example.
- Each task should be clear, concise, and understandable for an agent to execute or recognize.
- Avoid repetitive tasks; ensure each one is distinct.
- Consider a variety of household needs, including cleaning, maintenance, personal care, and organizational tasks.
- For each task, generate ten different objects or objects' combinations (e.g., [ObjectA, ObjectB] is a combination, [ObjectC, ObjectD, ObjectE] is another combination, [ObjectF] is a single object, [ObjectG, ObjectH] is another combination) that can satisfy the demand or be used to complete the task.
- Follow the format of the example provided in the Task Format.
- Each Object should be enclosed in double quotes, for example, "ObjectA".

Here are some language-grounding tasks:

task instruction: I want to exercise at home, focusing on my abdominal muscles, preferably with minimal noise.
basic solution: [exercise bike],[yoga mat, dumbbell]
preferred solution:[yoga mat]

task instruction: I need a secure way to store small valuables at home, preferably in a method that's not immediately obvious to guests.
basic solution: [safe],[bookcase, book]
preferred solution:[bookcase, book]

task instruction: I need an efficient method to keep track of my daily hydration, preferably with a solution that's easy to transport.
basic solution: [water bottle],[thermos]
preferred solution:[water bottle]

task instruction: I want to relax in my garden with a chair that gently rocks, but I prefer it to be able to withstand the weather conditions.
basic solution: [rocking chair, gazebo], [camp chair, umbrella]
preferred solution:[rocking chair, gazebo]

We can see that the language-grounding task uses some objects that do not appear in the HSSD dataset. We then generate several attributes for each task's instructions and objects.

We use the following prompts to generate attributes for instructions:

22

System Prompt: You are a household attribute generator. Generate a list of attributes or functions that can be used to satisfy the given instructions.

Prompt:
Attribute Introduction
Every object or item in the world has some attributes or functions that can be used to meet human demands. For example, water has the properties or functions of being drinkable, usable for washing clothes, capable of dissolving some solids, and suitable for bathing.
When a human need is satisfied by a certain object or item, it is essentially the attributes of that object that meet this demand. Therefore, we can deduce the required attributes or functions from a specific demand.

Attribute Format
String: List[String]
For example, "instruction": ["Attribute1", "Attribute2", "Attribute3", "Attribute4"]

Example
$Some Examples$

The Given Instruction
$Instruction$

Instructions
for each demand, generate four attributes or functions that can be used to satisfy the given demand, separated by "=============".
Instructions and Attribute should be separated by ": ".
Each attribute or function should be clear, concise, and understandable for an agent to execute or recognize.
Avoid using words that are too general or vague.
The attribute or function can be a word or phrase.
Avoid using synonyms or similar words that can be used interchangeably.
Consider a variety of attributes, including personal, organizational, and functional attributes.
Follow the Attribute Format provided above.
Maintain the order of The Given Instruction same with the original.

Here are instructions and their attributes.

task instruction: I want to exercise at home without assembling bulky equipment, preferring compact and versatile fitness tools.
attributes: [resistance bands, adjustable dumbbells, foldable yoga mat, doorway pull-up bar]

task instruction: I want to easily access my jewelry collection and minimize clutter, preferably by displaying the pieces.
attributes:[hanging organizers, jewelry stands, "drawer dividers, wall-mounted display]

task instruction: I want to keep my cosmetics organized and easily accessible, preferably in a tidy arrangement.
attributes:[compartmentalized storage, transparent containers, countertop design, cosmetic organizers]

task instruction: I want a simple way to track grocery needs, preferably one that the whole family can access.
attributes:[shared list, real-time updating, accessible by multiple users, simple interface]

We use the following prompts to generate attributes for objects:

System Prompt: You are a household attribute generator. Generate a list of attributes or functions that can be used to satisfy the given demand.

23

Prompt:
Attribute Introduction
Every object or item in the world has some attributes or functions that can be used to meet human demands. For example, water has the properties or functions of being drinkable, usable for washing clothes, capable of dissolving some solids, and suitable for bathing.
When a human need is satisfied by a certain object or item, it is essentially the attributes of that object that meet this demand. We can obtain the attributes and functions of an object from common sense or experience.

Attribute Format
String: List[String]
For example, Object1: ["Attribute1", "Attribute2", "Attribute3", "Attribute4"]

Example Object
water: ["drinkable", "usable for washing clothes", "capable of dissolving some solids", "suitable for bathing"]=============
apple: ["sweet", "edible", "vitamin C", "vitamin A"]=============
book: ["readable", "hold information", "hold data," "hold text"]=============
shirt: ["comfortable", "cloth", "wearable", "can be worn"]=============

Example Attribute
$ Example Attributes$

The Given Objects
$ Object Category$

Guidance
For each object, generate four attributes or functions that can be used to satisfy a demand, separated by "=============".
Object and Attribute should be separated by ": ".
Each attribute or function should be clear, concise, and understandable for an agent to execute or recognize.
Avoid using words that are too general or vague.
The attribute or functions can be a word or phrase.
Avoid using synonyms or similar words that can be used interchangeably.
Consider a variety of attributes, including personal, organizational, and functional attributes.
Follow ### Attribute Format ### provided above. Follow the content format of ### Example Attribute ###.
Maintain the Given object unchanged and only generate the attributes or functions.

Here are objects and their attributes.

object: long-handled duster
attributes:[extended reach cleaning, dust-attracting fibers, flexible head for tight spaces, washable and reusable]

object: rocking chair
attributes:[relaxation, comfortable seating, aesthetic addition to room, soothes babies]

object: adjustable table
attributes:[height customization, ergonomic design, collapsible for storage, multi-use surface]

object: dusting brush
attributes:[removes surface dust, gentle on delicate items, reaches tight spaces, ergonomic handle]

A.2.2 Attribute Training Details

The full loss function is shown below:

24

$$Loss = \lambda_1 \times Attribte\ Loss + \lambda_2 \times Matching\ Loss+$$
$$\lambda_3 \times VQ\ Loss + \lambda_4 \times Commit\ Loss + \lambda_5 \times Recon\ Loss \quad (6)$$

where λ_1 is 2.0, λ_2 is 1.0, λ_3 is 1.0, λ_4 is 0.25, and λ_5 is 1.0.

A.3 Details about Coarse-to-Fine Exploration

A.3.1 Design Details for Coarse Exploration Module

For the LLM branch, GPT-4 generates the attributes of the instruction in natural language. For example, "I need a comfortable place to read, preferably with natural light." has basic attributes of "comfortable seating", "quiet environment" and has preferred attributes of "natural light". Then, we use the CLIP-Text-Encoder to encode these language-level attributes into the instruction attribute features. For the MLP branch, we use CLIP-Text-Encoder to encode the instruction into the instruction features and then use Ins MLP Encoder to encode the instruction features into the instruction attribute features.

For the path-planning algorithm, we use the habitat-sim built-in greedy planner. We ensure each step is on an already explored point clouds to avoid navigating through unexplored points. In this paper, for a fair comparison, baselines and ablations that involve building maps and traveling on them will all use habitat-sim's built-in greedy planner, including but not limited to MOPA+LLM and FBE+LLM.

We provide some block score visualizations. A darker red color means a higher cosine similarity between the attribute features of the objects and the instruction. The darkness of the color is based on the rank of the block score among all blocks, not the actual score. Since scores are not calculated for blocks that have been visited before, only a single coarse exploration is included in the visualization. Note that the block with the highest score (*i.e.*, the darkest color) is likely to contain objects that can satisfy the demands. See Fig. 6, Fig. 7, Fig. 8, Fig. 9, Fig. 10. We also provide a visualization of block scores with different weights at Fig. 11.

A.3.2 Training Details about Fine Exploration Module

We use the standard transformer encoder from the official PyTorch 1.13.1 implementation, where d_model is 768, nhead is 8, num_layers is 6, and other parameters remain default. The embedding dim of action is 64. The embedding dim of GPS+Compass is 32. The input dim of LSTM is 768+64+32, its hidden_size is 1024, and its num_layers is 2. The depth model is a simple five-layer CNN model and a two-layer MLP model.

We collect about 50,000 trajectories to train the fine exploration module under seen tasks and seen scenes settings in Experiments Sec. 5 using imitation learning. We collect trajectories by following steps: 1) randomly select a scene and a task, 2) initialize agent within two meters (*i.e.*, block size in coarse exploration) of a target object, 3) use habitat-sim's built-in greedy planner to get the next step, 4) when the distance to the target object is less than 0.2 meters, turn left/right or look up/down according to the height and position of the object, 5) when the target object is in the field of view, execute $Find$ and close this trajectory. The trajectories we collect here are relatively short and end with only one execution of $Find$. The average length of these trajectories is 9.19. This is because the fine exploration module only needs to take on exploration within a block, and therefore, the goal of learning only needs to be a trajectory within a block. Training with such short trajectories is very efficient and requires few computational resources (only a RTX 4090 graphics card is needed). The standard deviation of the trajectory length is 5.62. The median trajectory length is 8. The plurality of trajectory lengths is 3. The maximum trajectory length is 51. The minimum trajectory length is 2. We trained the model on a single RTX 4090 using imitation learning and cross-entropy loss, i.e., considering the action prediction as a classification task, consuming about 12h.

We keep switching between the two exploration phases until the number of $Find$ reaches n_{find}, and then execute $Done$. When $Find$ is executed in the fine exploration phase, switch back to the coarse exploration phase to continue selecting the next waypoint.

25
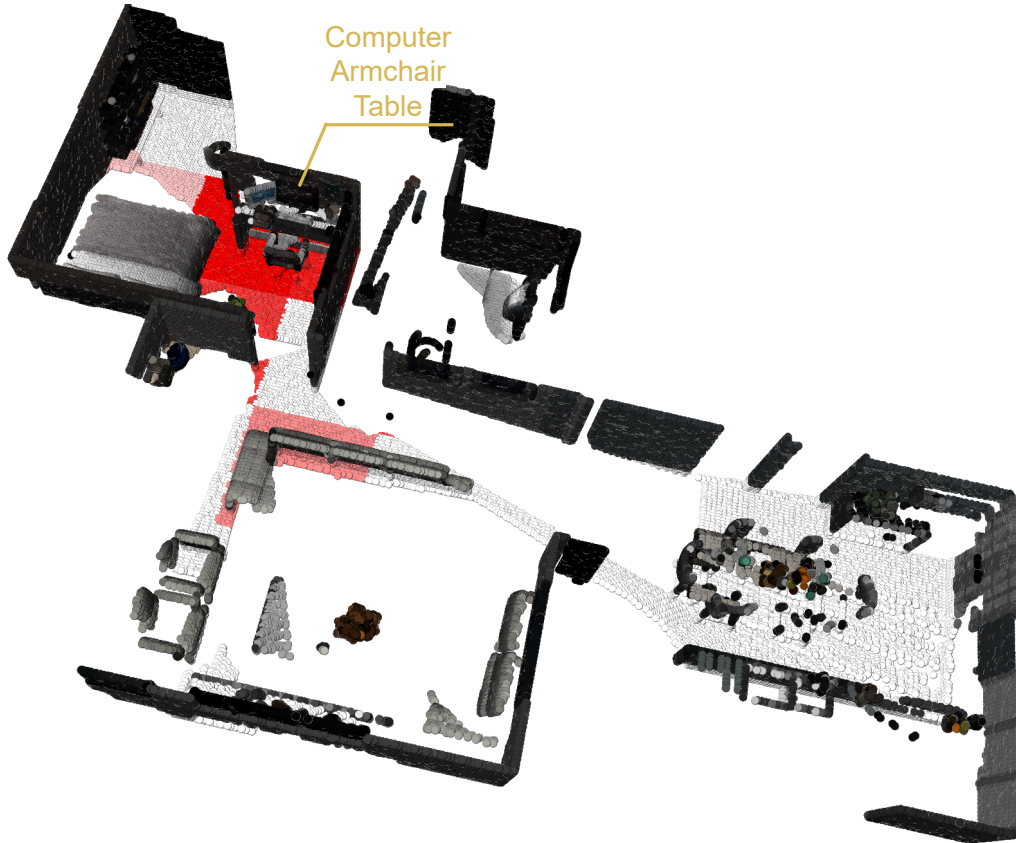
Figure 6: **Block Visualizations.** Instruction: I need to take quick notes during a meeting, preferably with a device that saves them digitally. Solution: Computer, Armchair, Table.

A.4 Experiments

A.4.1 Details about Experimental Settings

The effect of each action is as follows:

- MoveAhead: Move forward 0.25 meters.
- RotateRight: Turn right 30 degrees.
- RotateLeft: Turn left 30 degrees.
- LookUp: Turn the camera up 30 degrees.
- LookDown: Turn the camera down 30 degrees.
- Find: Record objects with a distance less than d_{find} in the current field of view to the found list.
- Done: End the current episode and report the success rate and SPL.

Note that we want to focus the benchmark on navigation. When executing Find, the simulator will automatically add objects to the found list that have distances less than d_{find} in the current field of view, regardless of whether or not they are recognized by the object detection module GLEE. Such detection is done by the simulator, so the agent does not get the ground truth semantic label, except when Find determinations are made by MOPA+LLM and FBE+LLM.

Our method and baselines can be trained on a single RTX 4090, which will take about one day for each method. We report the mean and standard deviation for at least three seeds. We test the agent for 100 epochs in each seed, which takes about 8 hours.

26
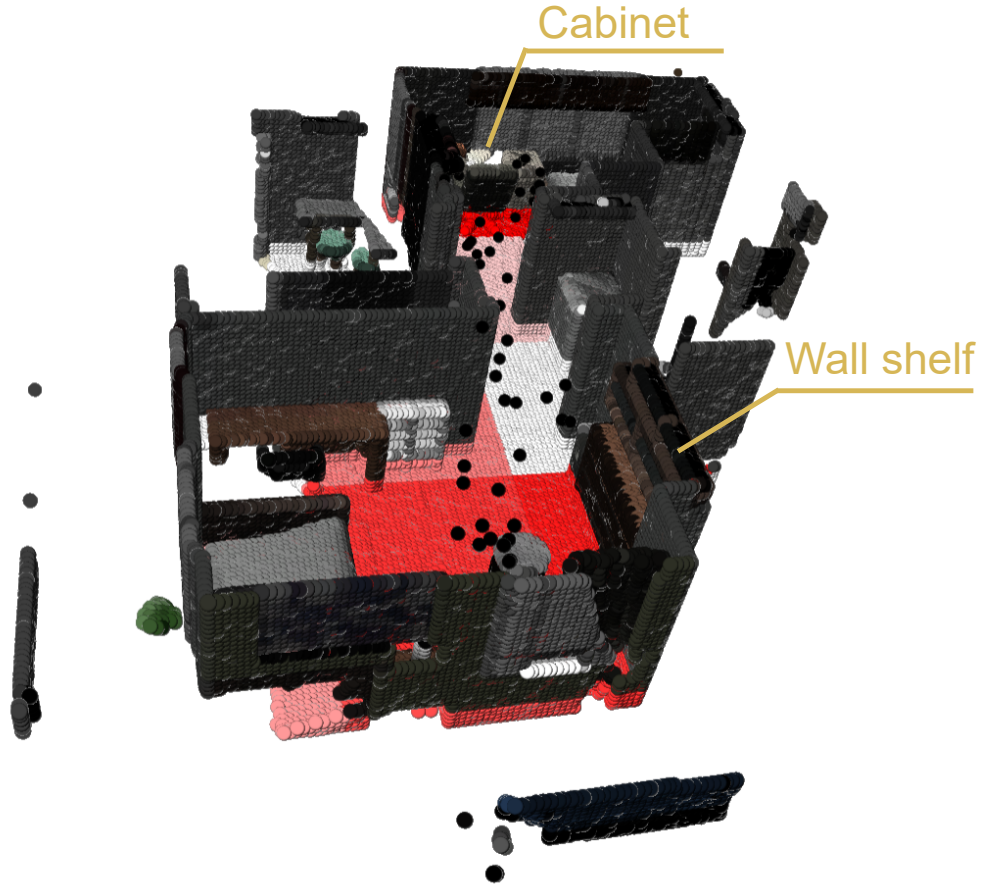
Figure 7: **Block Visualizations.** Instruction: I need to store a collection of fine china, preferably in a way that displays them elegantly. Solution: Wall shelf, Cabinet.

A.4.2 Details about Baselines

Since many baselines are policies in a single object navigation setting, we make some modifications to their action space. In their original policy, Done is modified to Find, and Done is automatically output when the number of Find executions reaches the upper limit n_{find}.

For Random, we let the agent randomly select an action other than Done and execute it. When the number of Find reaches the maximum number of executions n_{find} or the number of steps reaches the limit, execute Done.

For VTN, it is trained in the original paper in two stages, imitation learning without LSTM and reinforcement learning with LSTM. We remove the reinforcement learning phase here because we find that in the MO-DDN task, using reinforcement learning to train VTN leads to performance degradation. We, therefore, chose to collect 20,000 trajectories and train the VTN using imitation learning with an LSTM module. We collect trajectories by following these steps: 1) randomly select a scene and a task, 2) initialize agent at a random position, 3) set a target object according to the solution, 4) use habitat-sim's built-in greedy planner to get the next step, 5) when the distance to the target object is less than 0.2 meters, turn left/right or look up/down according to the height and position of the object, 6) when the target object is in the field of view, execute Find, 7) go back to step 3 until all objects in the solution have been searched for, and then execute Done. The average length of these trajectories is 49.15. We replace the VTN's original goal description with the CLIP features of the demand instruction. During evaluating, the agent executes Done when the number of Find reaches the maximum number of executions n_{find} or the number of steps reaches the limit.

27
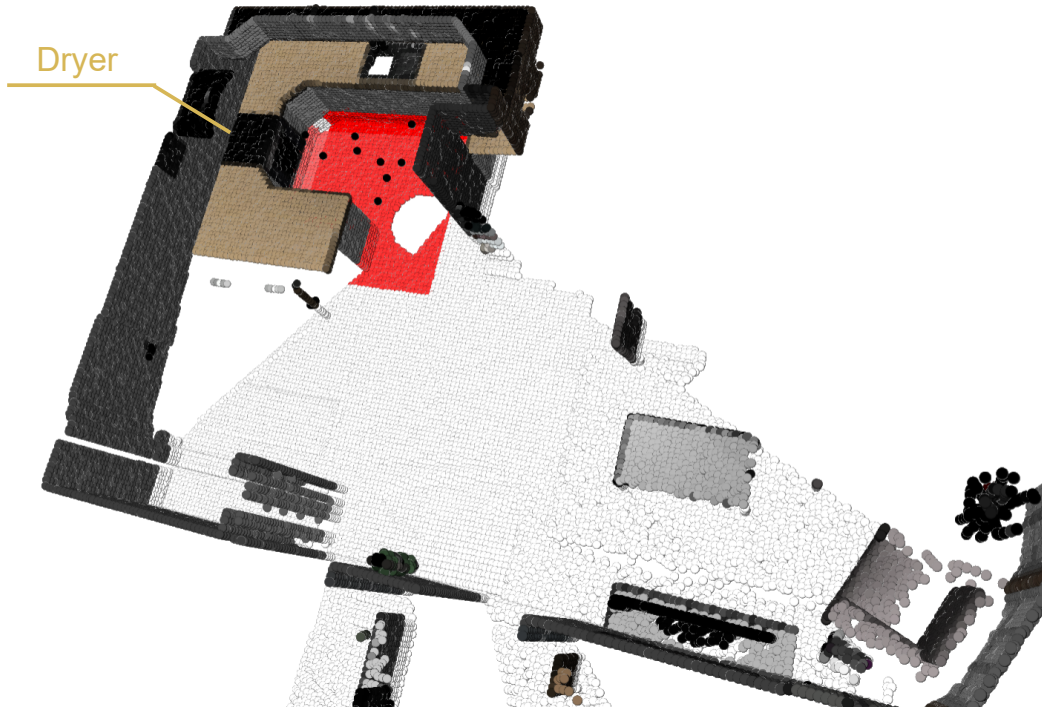
Figure 8: **Block Visualizations.** Instruction: I need to quickly dry a batch of laundry, but I prefer an fast and energy-efficient method. Solution: Dryer.

For ZSON, we use imitation learning to fine-tune the official ZSON's trained weights using the trajectories collected in the VTN. We take the CLIP features of the demand instruction as ZSON's goal descriptions. During evaluating, the agent executes Done when the number of Find reaches the maximum number of executions n_{find} or the number of steps reaches the limit.

For DDN, we train the agent using the trajectories collected in the VTN. For training the attribute model in DDN, we concatenate all objects in a solution at the language level and encode them with CLIP-Text-Encoder as object features. If two solutions satisfy the same demand instruction, then the object features corresponding to these two solutions are positive samples. During evaluating, the agent executes Done when the number of Find reaches the maximum number of executions n_{find} or the number of steps reaches the limit.

For MOPA+LLM, we use map-building and object-detection models that are the same as our method, and we use LLM to make selections for currently recognized objects. When the LLM selects an object as a target, we use habitat-sim's built-in greedy planning to walk near it; if the LLM does not select any object, it chooses a random point on the already explored map. Compared to the original MOPA implementation, we replace the point goal agent with the built-in greedy planner of habitat-sim. When the agent reaches the target, we prompt the LLM to decide whether to perform the Find action. The prompts we use are in Sec. A.4.4.

For FBE+LLM, we use the FBE as the exploration module and then let the LLM choose which objects that have been explored meet the demand and serve as targets for the habitat-sim's built-in greedy planner to walk around. If the LLM does not select an object, the agent continues to explore using the FBE method. When the agent reaches the target, we prompt the LLM to decide whether to perform the Find action. The prompts we use are in Sec. A.4.4.

A.4.3 Details about Ablation Study

Ablation on Coarse Exploration   For FBE+Fine, we use FBE to select waypoints and then use the habitat-sim's built-in greedy planner to get to the waypoints. We use the fine exploration module to locate the target objects when arriving at the waypoints. For LLM+fine, we use LLM to select

28
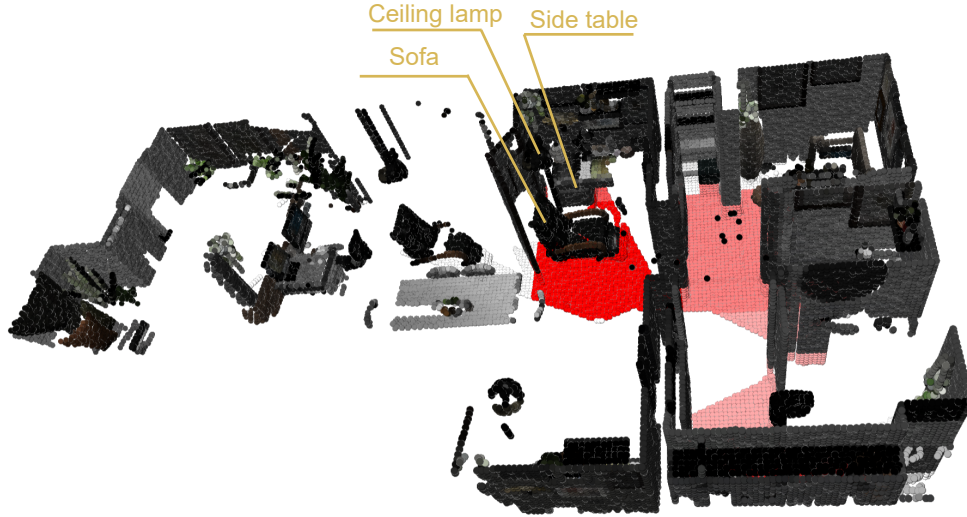
Figure 9: **Block Visualizations.** Instruction: I need to find a comfortable place to read for my study group, preferable with good lighting. Solution: Sofa, Side table, Ceiling lamp.

waypoints, and everything else is the same as FBE+Fine. The prompts we use are in Sec. A.4.4. For CLIP+fine, we replace attribute features with CLIP features to compute cosine similarity as block scores.

Ablation on Fine Exploration For Coarse+ZSON/VTN, we use the same trajectory dataset used to train the fine exploration module to train ZSON and VTN and then replace the fine exploration module with ZSON/VTN. For Coarse+Random, we replace the fine exploration module with randomly selecting an action except for Done.

Ablation on Attribute Training For Ours w/o VQ-VAE, we set the weight of VQ Loss, Commit Loss, and Recon Loss to zero and only maintain Attribute Loss and Matching Loss. For Ours w/o initialization, we do not load the clustering centers to codebook.

Ablation on Score Weights We modify the attribute query to adjust the weighting of the basic and preferred scores. See 3 for the exact formula.

A.4.4   Details about LLM's Prompt in Experiments

In our baseline experiments and ablation study, we use LLM to select the waypoint (*e.g.*, MOPA+LLM, FBE+LLM, and LLM+Fine) and decide whether to perform the Find action or not (*e.g.*, MOPA+LLM and FBE+LLM).

We use the following prompts to let LLM select the waypoint:

> System Prompt: You are an AI assistant that can understand human demands and imagine what human demands can be met with some combinations of objects. Now, I need you to assist in selecting positions on the map based on current location and recorded object locations.

29
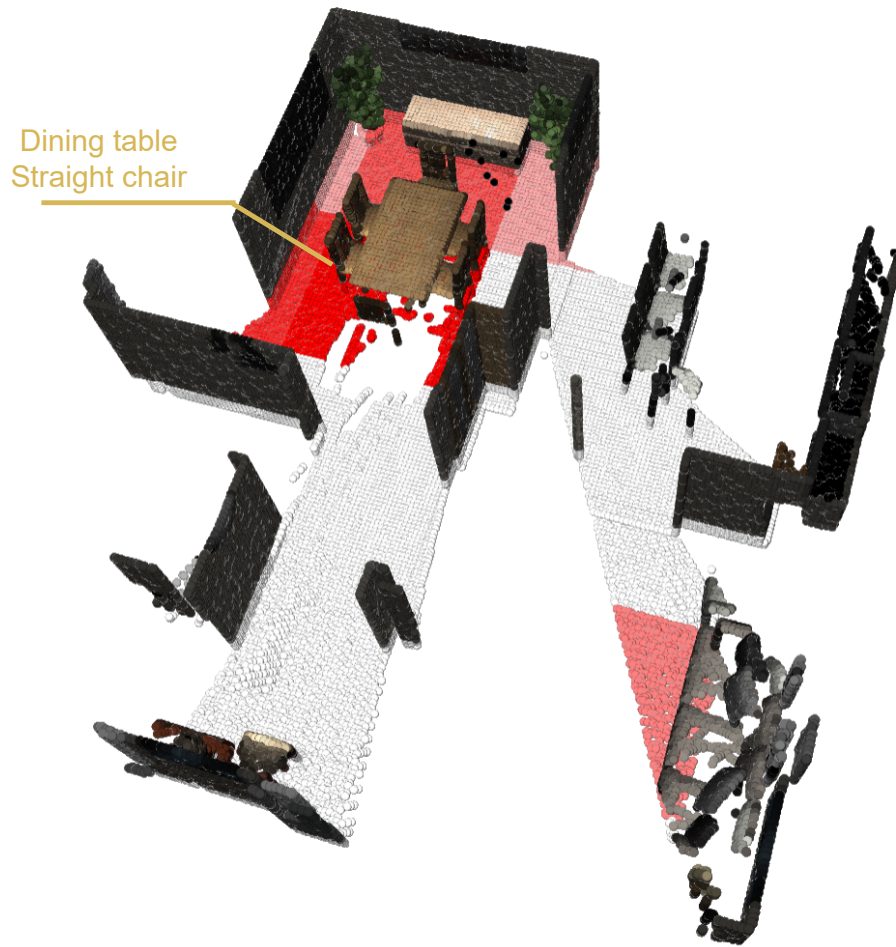
Figure 10: **Block Visualizations.** Instruction: I need to organize a small evening gathering but want to avoid any accidents with real candles. Solution: Dining table, Straight chair.

Prompt:
Task
Now, I hope you can help me more efficiently select areas in the scene for exploration using your knowledge. Specifically, I will provide you with the current records of objects available for exploration, along with their names, positions, and distances in ### Explored Object List###. You need to choose one of them and proceed to explore the area nearby (if you determine that none of the objects on the ### Explored Object List ### are suitable, you can choose to output FBE, and I will use Frontier Base Exploration in this round). My goal is to find a set of objects that can complement each other to meet my demands. I will give you the objects I have already found (in ### Found Object List ###), and your task is to consider the objects already found (if not empty) and efficiently search for the missing ones to meet my requirements.
Demand Instruction
$Instruction$
Found List
$Found List$
Explored Object List
$Object in Point Clouds$

30

$$r_b = 1 \; and \; r_p = 0 \qquad\qquad r_b = 1 \; and \; r_p = 2$$
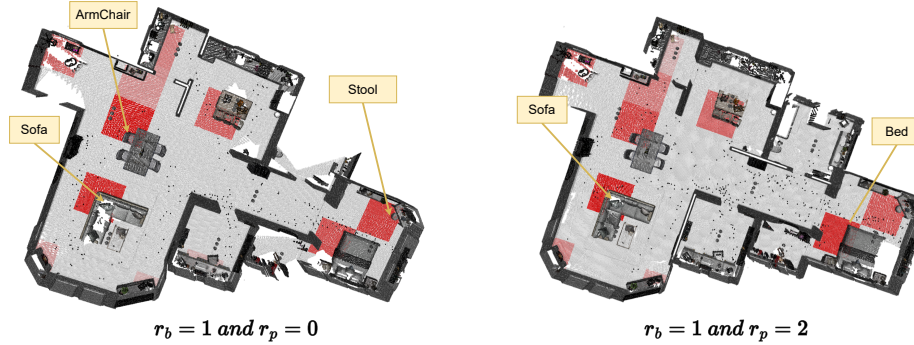
Figure 11: **The Effect of** r_b **and** r_p. This is an example used only to demonstrate scoring blocks. The darker the color in the figure, the higher the block score. When the task is "Find a place to sit, preferring a comfortable and soft place", lowering r_p (left) ignores preferences, so ArmChair, Sofa and Stool can get a high score (dark red); raising r_p (right) focuses on preferences, so Sofa and Bed can get a high score.

Reply Format
"I choose: {obj_name}"
The "" placeholder must be included in the response; it is used to identify the answer. You must strictly output the names from the ### Explored Object List ###; you cannot invent other objects or provide multiple objects. For example, if you think the most probable object to explore next is "apple", you should output "I choose: apple". If you think none of the objects in the list are suitable, you can choose to output "I choose: FBE"
Requirement
You need to carefully consider this: In order to meet my demand, besides the objects mentioned in ### Found Object List ###, what other types of objects need to be searched for, and which of these objects are most likely to appear near an object in the ### Explored Object List ###. In the end, I only need you to output the name of the most probable object from the ### Explored Object List ### as the area you should prioritize exploring next. Please strictly follow the format I declared in ### Reply Format ###. Please think carefully and select the relatively best object as the next exploration target. Only when you can't think of any good choices should you output FBE.

We use the following prompts to let LLM decide whether to perform the Find action or not:

System Prompt: You are an AI assistant that can understand human demands and imagine what human demands can be met with some combinations of objects.

31

Prompt:
Task
Surrounding Object List ### declares the objects currently around you. You need to carefully inspect them to see if there are any that are relevant to the demand I stated in ### Demand Instruction ###, and that can be used in conjunction with previously found objects (in ### Found Object List ###). If any of the objects in ### Surrounding Object List ### might be potentially useful, report "Find". If there are no objects that could be useful or relevant, report "Skip".
Demand Instruction
$Instruction$
Found List
$Found List$
Surrounding Object List###
$ground truth semantic label in the current field of view $
Reply Format
I will execute $Action$.
You should replace the Action with Find or Skip, "$$" is an identifier used to specify the range of the answer; please make sure you will not use it in other places.
Requirement
Exploration steps reflect the progress of exploration. In total, you can perform max_step exploration steps, and currently, $current_step$ steps have been completed. You can execute 'Find' a total of 5 times, and so far, you have executed it $current_find_time$ times. Please consider both the progress of exploration and the number of explorations performed to decide whether to execute 'Find' or 'Skip' now. Feel free to use your imagination and try to find any object that might be potentially useful. Don't waste your find opportunities.
You can take your time to consider whether the surrounding objects can help me meet my demand and also think about which objects among the surrounding ones can complement those already found. However, you must strictly use the answer format declared in ### Reply Format ###.

A.5 More Limitations

In our method, we do not let the agent decide to choose Done until the number of Find reaches a threshold or the number of steps reaches a threshold. This leads to the possibility that the objects in the found list already fully satisfy the demand instruction, yet the agent will still enter the coarse exploration module to select the next waypoint, resulting in a low SPL value. Future work could explore the attribute features used for decision-making in Done.

32

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and precede the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We demonstrate the conclusions in the Experiment Section 5.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss limitations in the Conclusion Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.

33

- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: This paper does not have theoretical results

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: A full description of the model and prompts for large language models are provided in our main paper 5 and supplementary material A.4. We will release all the code and dataset if this paper is accepted.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.

34

- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.

- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example

  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: But we will release all the code and dataset if this paper is accepted.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

35

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: Please see the main paper 5 and supplemental material A.4.1.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We report the standard deviation of results in Tab. 5.3.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: A single RTX4090 is enough to train and test our agent. It takes one day to train the agent and two days to test. Please see the supplemental material for details A.4.1.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.

36

- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: We do.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: by [Yes]

Justification: We discuss them in the Conclusion Section 6.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: We focus on navigation. To the best of our knowledge, it does not have a high risk.

Guidelines:

- The answer NA means that the paper poses no such risks.

37

- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: We follow the protocol for the dataset.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We provide some examples of our task dataset in the supplemental material A.1.1.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: the paper does not involve crowdsourcing nor research with human subjects.

38

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: the paper does not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

39