Collaborative Instance Object Navigation: Leveraging Uncertainty-Awareness to Minimize Human-Agent Dialogues

Supplementary Material

In this supplementary material, we first provide additional details regarding the CoIN-Bench dataset (Sec. A), including an overview of the GOAT-Bench dataset on which CoIN-Bench is based, as well as statistics and examples of target instances within CoIN-Bench. Additionally, in Sec. B, we provide a visualization of the evaluation setup, clarifying the role of the user responses in our evaluation, whether from real human or simulation. Next, we elaborate on the implementation details for baseline comparisons in Sec. C, and present a computational analysis of AIUTA in Sec. D. Further details on the real human experiments are provided in Sec. E. Then, in Sec. F we investigate the low performance of the original VLFM, and in Sec. G, we provide a UMAP visualization of the questions generated by the agent. Next, in Sec. H, we detail the evaluation conducted on IDKVQA including the dataset creation, evaluation metric, and state-of-the-art baselines used for comparison. Finally, we include all the prompts in Sec. I and the full algorithm of AIUTA in Sec. J.

For a demonstration of AIUTA in action, engaging with a real human through natural language dialogues to collaboratively localize a target instance, please refer to the accompanying video (**aiuta_demo.mp4**) provided in the supplementary material.

A. Additional details of CoIN-Bench

A.1. CoIN-Bench

Instance examples. The CoIN-Bench benchmark poses a significant challenge, since multiple distractor objects are present among each target. To illustrate this, Fig. 1 provides examples where the target instance is highlighted with red borders, while distractors in the same scene are marked with blue borders. As demonstrated, agent-user collaboration is crucial to gather the necessary details for uniquely identifying the target instance among other visually similar objects of the same category, such as the armchair or the plant.

Dataset statistics. We provide additional statistics for the CoIN-Bench dataset. In Fig. 2 we show the shortest path statistics for the CoIN-Bench dataset. In particular, the euclidean and geodesic distance for all the split, as well as the number of distractors. Next, Fig. 3 illustrates the distribution of instance categories across different splits. These splits are ordered by dataset size, from the largest at the top (`Val Seen`) to the smallest at the bottom (`Val Seen Synonyms`). The number of distinct categories decreases as the dataset size reduces. The `Val Seen` split, being the largest, also contains the highest number of distinct categories, with "cabinet", "bed", and "table" being the top 3 common categories. `Val Seen Synonyms`, being the smallest, only contains 3 categories.

A.2. GOAT-Bench

Dataset. GOAT-Bench provides agents with a sequence of targets specified either by category name c (using episodes from [64]), language description d, or image in an open vocabulary fashion, using the HM3DSem [43] scene datasets and Habitat simulator [48]. Natural-language descriptions d are created with an automatic pipeline by leveraging ground-truth semantic and spatial information from simulator [48] along with capabilities of VLMs and LLMs. Specifically, for each object-goal instance, a viewpoint image is sampled to maximize frame coverage. From this sampled image, the names and 2D bounding box coordinates of visible objects are extracted. Then, spatial information is extracted with the BLIP-2 [22] model, while ChatGPT-3.5 is prompted to output the final language description.

Splits. GOAT-Bench baselines are trained on `Train` split, and evaluated on validations splits. Notably, the evaluation splits are divided into `Val Seen` (*i.e.*, object categories seen during training), `Val Seen Synonyms` (*i.e.*, object categories that are synonyms to those seen during training) and `Val Unseen` (*i.e.*, novel object categories).

B. Evaluation Setup

In Fig. 4, we show the two evaluation setups, highlighting their differences between the human user and the simulated user-agent interactions. Fig. 4 (Left) shows how a human user answers the agent's queries based on their knowledge of the target instance. However, relying on human responses for large-scale evaluations is impractical due to variability, scalability constraints and large cost. To address this, we introduce a simulated user-agent interaction setup as in Fig. 4 (Right). The user responses are simulated via a VLM with access to the high-resolution target instance image, which is never available to the agent. With the visual coverage of the target instance, the simulated user responses can support the diverse open-ended, template-free questions from the agent, about any attribute of the target instance. This is more desired than previous work [9] whose simulation setup leverages an LLM with access to the instance description, particularly when the instance description misses critical fine details that the agent deems important to know. For instance, in the case of the picture in Fig. 1 of the main paper,

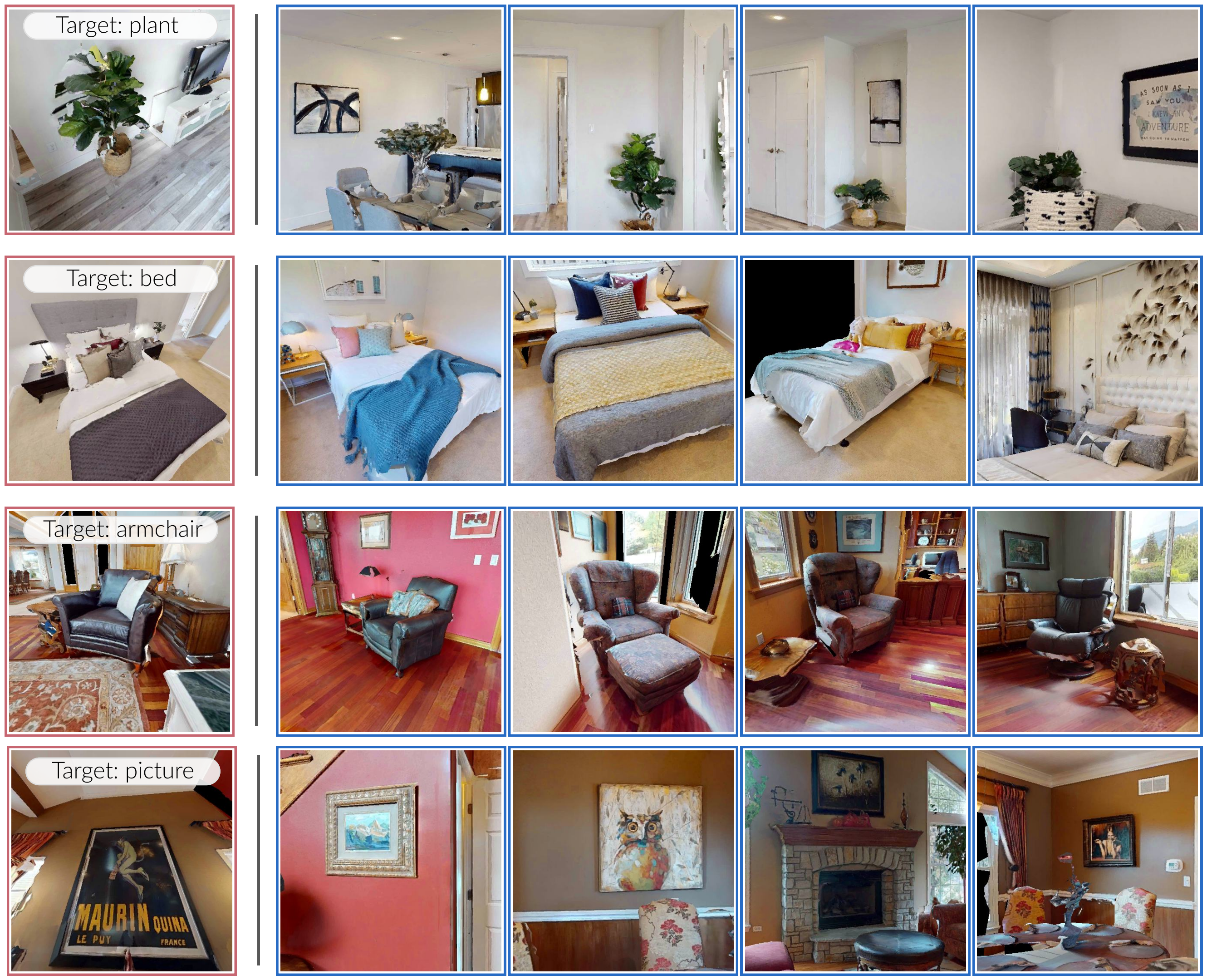


Figure 1. CoIN-Bench can be very challenging when only given the instance category to the agent. We highlight the target instance with red borders, while the distractor instances that exist in the same scene are marked with blue borders.

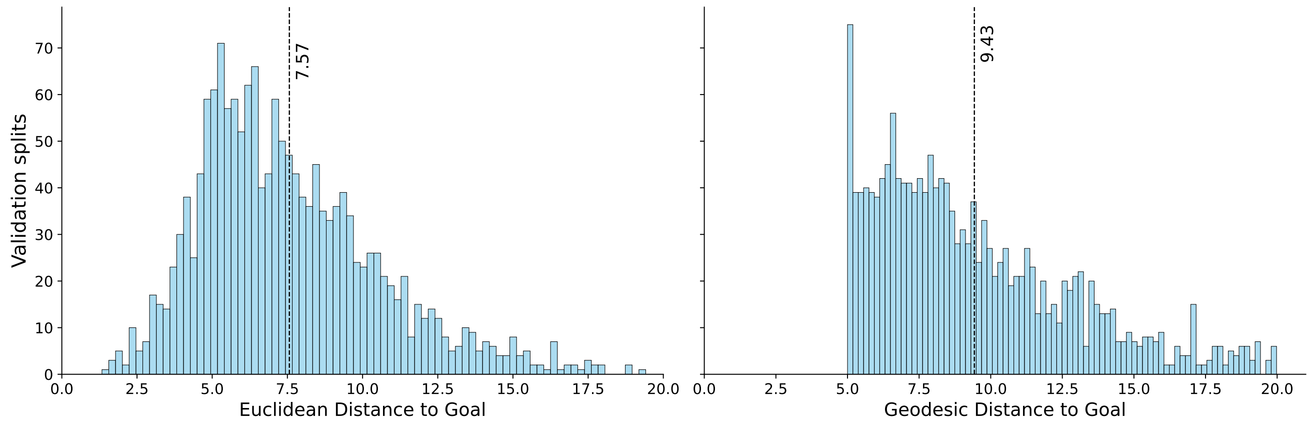


Figure 2. Distribution of the path length to the goal, both in Euclidean and Geodesic term.

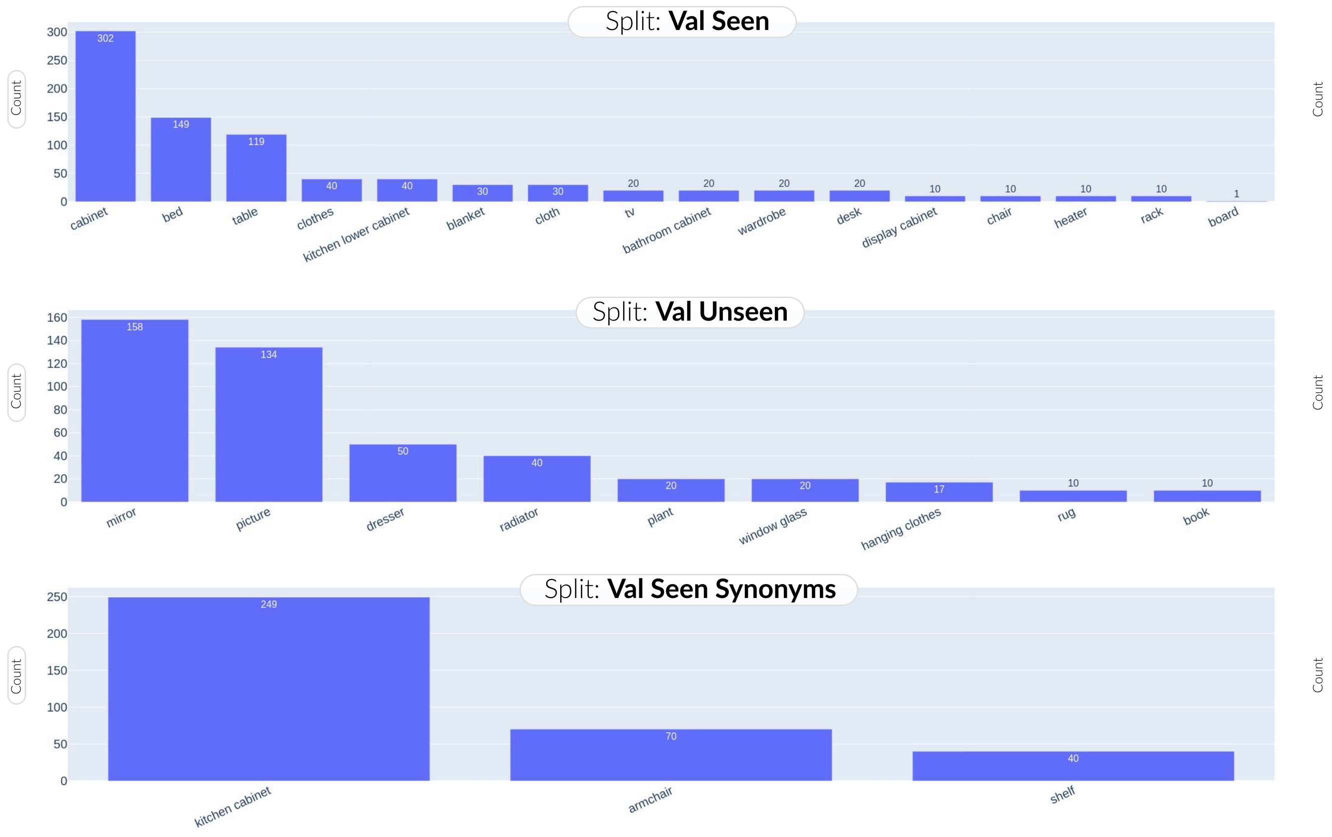


Figure 3. We show the distribution of categories, categorized for each evaluation split.

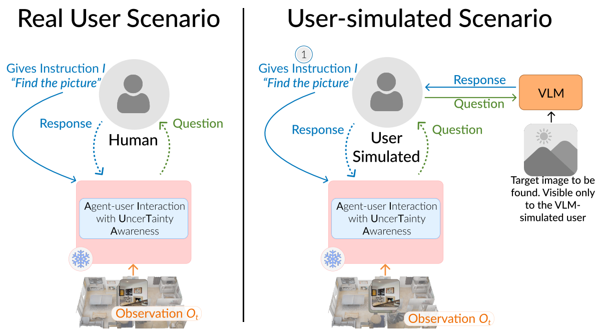


Figure 4. CoIN-Bench evaluation setup. (Left) Real human responding to the agent's question. (Right) Simulated user-agent interactions, where the user responses are provided by a VLM with access to a high-resolution target instance image for scalable and reproducible experimentation.

the instance description may not mention "the person is shirtless", but this detail is critical for the agent to eventually disambiguate the target instance from distractors.

C. Baselines

In this section, we provide a description of the different baselines for Instance Navigation and Object Navigation used throughout the paper: VLFM (Sec. C.1), Monolithic (Sec. C.2), PSL (Sec. C.3), and OVON (Sec. C.4).

C.1. VLFM

VLFM [63] is a zero-shot state-of-the-art object-goal navigation policy that does not require model training, pre-built maps, or prior knowledge about the environment. The core of the approach involves two maps: a frontier map (see in Fig. 5 (a)) and a value map (see Fig. 5 (b)).

Frontier map. The frontier map is a top-down 2D map built from depth and odometry information. The explored area within the map is updated based on the robot's location, heading, and obstacles by reconstructing the environment into a point cloud with the depth images, and then projecting them onto a 2D grid. The role of the frontier map is to identify each boundary separating the explored and unexplored areas, thus identifying the frontiers (see the blue dots in Fig. 5 (a)).

Value map. The value map is a 2D map similar to the frontier map. For each point within the explored area, a value is

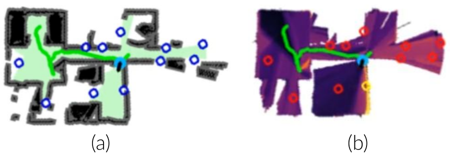


Figure 5. (a) Frontier map and (b) value map constructed by VLFM [63]. The blue dots in (a) (as well as the red dots in (b)) are the identified frontiers.

assigned by quantifying its relevance in locating the target object (see Fig. 5 (b)). At each timestep, frontiers are extracted from the frontier map, and the frontier with the highest value on the value map is selected as the next goal for exploration. To efficiently guide the navigation, VLFM projects the cosine similarity between the current visual observation and a textual prompt (*e.g.*, *"Find the picture"*) onto the value map. This similarity is computed using the BLIP-2 model [22], which achieves state-of-the-art performance in image-to-text retrieval. To verify whether a target instance is present in the current observation, VLFM employs Grounding-DINO [27], an open-vocabulary object detector. Once a candidate target is detected, Mobile-SAM [67] refines the detection by segmenting the object's contour within the bounding box. The segmented contour is paired with depth information to determine the closest point on the object relative to the agent's position. This point serves as a waypoint for the agent to navigate toward the object.

At each timestep, the action a_t is selected using a PointGoal navigation (PointNav) policy [2], which can navigate to either a frontier or a waypoint, depending on the context.

C.2. Monolithic

The Monolithic (SenseAct-NN Monolithic Policy) is a single, end-to-end reinforcement learning (RL) policy designed for multimodal tasks, leveraging implicit memory and goal encoding proposed in [19]. RGB observations are encoded using a frozen CLIP [42] ResNet50 encoder. Additionally, the agent integrates GPS and compass inputs, representing location $(\Delta x, \Delta y, \Delta z)$ and orientation $(\Delta\theta)$. These inputs are embedded into 32-dimensional vectors using a encoder with fully connected layers. To model multimodal inputs, a 1024-dimensional goal embedding is derived using a frozen CLIP image or CLIP text encoder, depending on the subtask modality (object, image, or language). All input features—image, location, orientation, and goal embedding—are concatenated into an observation embedding, which is processed through a two-layer, 512-dimensional GRU. At each timestep, the GRU predicts a distribution over a set of actions based on the current observation and the hidden state. The policy is trained using 4×A40 GPUs for approximately 500 million steps.

C.3. PSL

PSL [52] is a zero-shot policy for instance navigation, which is pre-trained on the ImageNav task and transferred to achieve object goal navigation without using object annotations for training.

Built on top of ZSON [31], observations are processed by a learned ResNet50 encoder and a frozen CLIP encoder obtaining, respectively, observation embeddings and semantic-level embeddings. To encode the goal modality, an additional frozen CLIP encoder is used, obtaining goal embedding. The goal and the semantic-level embeddings are additionally processed by a semantic perception module, which reduces dimension condensing critical information, emphasizing the reasoning of the semantics differences in the goal and observation. Based on condensed embeddings and observation embeddings, the authors trained a navigation policy using reinforcement learning. Specifically, the PSL agent is trained for 1G steps following ZSON [31], on 16 Nvidia RTX-3090 GPUs.

C.4. OVON

OVON [64] is a transformer-based policy designed for the open-vocabulary object navigation task. At each timestep t, it constructs a 1568-dimensional latent observation o_t of the current navigation state by concatenating the encodings of the current image I_t, object description c, and previous action a_{t-1}, using two SigLIP encoders [66] and an action embedding layer. This latent observation is then passed to a 4-layer decoder-only transformer [59], which, along with the previous 100 observations, outputs a feature vector. This vector is then used to produce a categorical distribution over the action space via a simple linear layer. The policy is trained on their proposed HM3D-OVON dataset using various methods, such as RL and BC, for 150M to 300M steps, across 16 environments on $8\times$ TITAN Xp, resulting in a total of 128 environments. Note that the categories seen in training in HM3D-OVON overlap with that of GOAT-Bench.

D. Computational analysis

In the following, we report the average inference time of AIUTA over 20 episodes, using an NVIDIA 4090 GPU, following the steps outlined in the algorithm in Sec. J.2: *Step* 1, Detailed Detection Description: 11.3s; *Step 2,* Perception Uncertainty Estimation: 8.29s; Step 3 + Interaction Trigger: 6.13s. We would like to emphasize that our code was not optimized for speed, as it is out of scope of our study- we did not apply model compilation (*e.g.*, `torch.compile`) and quantization, leaving room for further efficiency improvements. Moreover, in AIUTA, we identify the primary bottleneck is the LLM call. As discussed in the *Conclusion* (Sec. 7) and in [8], reducing model dimensionality while maintaining similar reasoning performance is a necessity and a promising direction for future work. Additionally, our inference time is in line with other works [8]. Finally, the emerging field of Language Processing Units (LPUs) offers potential solutions, promising near-instant inference, high affordability, and energy efficiency at scale [14].

E. Evaluation with real human

To demonstrate the reliability and reproducibility of our simulated setup, we run a human study comparing the performance of AIUTA when user responses are provided by: *(i) real human* and *(ii) simulations* (Fig. 4). A total of 20 volunteers participated in the study (12 males and 8 females), with ages ranging from 20 to 40 years. All participants have backgrounds in electronic engineering, computer science, or other relevant fields, minimizing expertise barriers to conducting the experiments. At the start of each episode, participants are given an image depicting the final target instance, which remains accessible throughout the experiment. Again, note that this image is never seen by the agent. This setup simulates a real-world scenario where a human has a reference image in mind, enabling them to answer questions correctly. The human user then initiates the navigation by sending the initial instruction to the agent (using the fixed template "`Find the <category>`") via a chat-like User Interface (UI) that we have developed for the evaluation (as demonstrated in the supplementary video, **aiuta_demo.mp4**). Next, the human user is encouraged to respond to the questions posed by AIUTA in natural language and to truthfully reflect the facts about the target instance. For this evaluation, we have selected 40 episodes across CoIN-Bench dataset, randomly distributed among participants, with each conducting two evaluations. When compared with the simulated setting, we found no statistical differences in terms of results, showing that our simulated-based evaluation is reliable and reproducible.

F. VLFM results

In this section, we investigate the low `SR` results of VLFM in Tab. 2 of the main paper. To better understand this behavior, we introduce an additional metric, `Distractor Success`, which mirrors the success rate but considers an episode successful if the agent stops at a distractor object instead of the target. As we can see in Tab. 6, VLFM successfully locates the correct category instance (`Distractor Success)` but struggles to discern its attributes and differentiate between instances (low `SR)`. Furthermore, this analysis highlights that the presence of sufficient distractors is well realized with our dataset construction procedure.

G. UMAP visualization

To illustrate the diversity of questions to the user generated by AIUTA, we collect 414 question samples made by the agent, compute embedding using Sentece-Bert [46] and visualize them using UMAP [33] for dimensionality reduction. The results, shown in Fig. 6, demonstrate that AIUTA generates questions covering a wide range of attributes, such as color, material, style, and spatial arrangement.

Statistics	Val Seen	Val Seen Synonyms	Val Unseen
`SR`	0.36	0.00	0.00
`Distractor Success`	3.37	0.84	4.58

Table 6. `SR` and `Distractor Success` comparison.

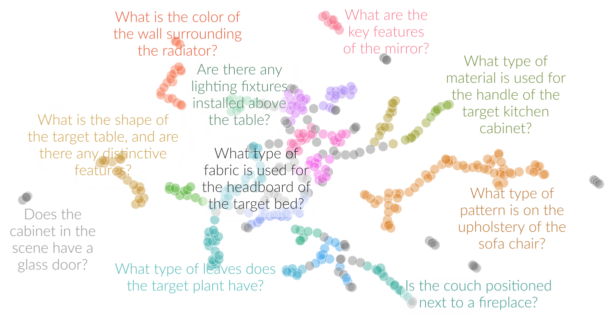


Figure 6. AIUTA generates questions covering a wide range of attributes, such as color, material, style, and spatial arrangement.

H. IDKVQA dataset

H.1. Dataset

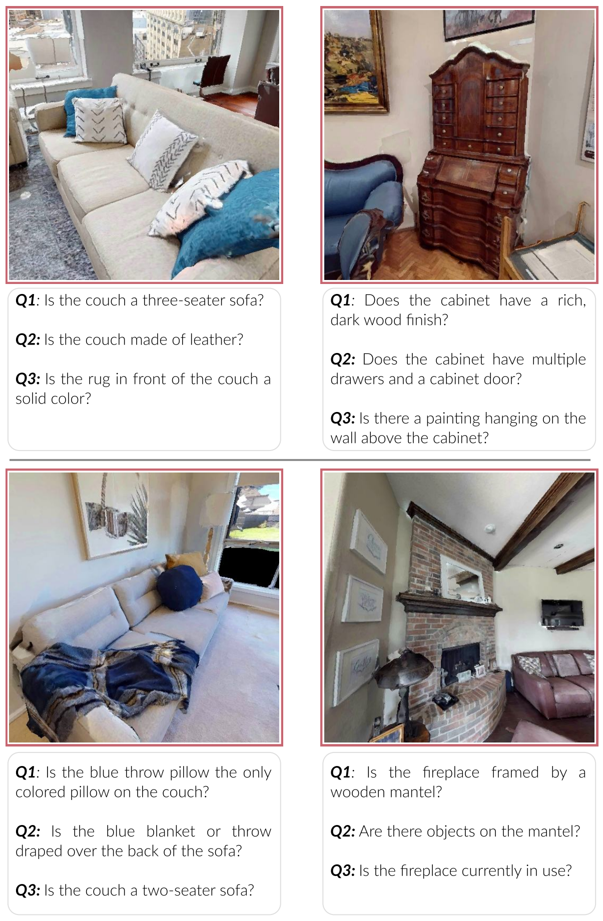
An essential feature of the *Self-Questioner* is its ability to generate self-questions aimed at extracting additional attributes from the observation O_t and assessing the uncertainty of the VLM. However, there exists no dataset in the such context for us to understand if how reliable a technique is for the VLM uncertainty estimation.

For this purpose, we introduce IDKVQA, a dataset specifically designed and annotated for visual question answering using the agent's observations during navigation, where the answer includes not only `Yes` and `No`, but also `I don't know`. Specifically, we sample 102 images from the training split of GOAT-Bench. Then, for each image, we leverage the *Self-Questioner* pipeline to generate a set of questions. Each question is annotated by three annotators, that can pick one answer from the set {`Yes, No, I don't know`}. Fig. 7 illustrates sample images and their questions generated by the *Self-Questioner* module.

H.2. VLM uncertainty estimation on IDKVQA.

In this section, we present a detailed analysis of VLM uncertainty estimation on IDKVQA, focusing on the evaluation metric and baseline methods.

Metric. We evaluate the performance using the *Effective Reliability* metric Φ_c proposed in [60]. This metric captures



Q1: Is the couch a three-seater sofa?

Q2: Is the couch made of leather?

Q3: Is the rug in front of the couch a solid color?

Q1: Does the cabinet have a rich, dark wood finish?

Q2: Does the cabinet have multiple drawers and a cabinet door?

Q3: Is there a painting hanging on the wall above the cabinet?

Q1: Is the blue throw pillow the only colored pillow on the couch?

Q2: Is the blue blanket or throw draped over the back of the sofa?

Q3: Is the couch a two-seater sofa?

Q1: Is the fireplace framed by a wooden mantel?

Q2: Are there objects on the mantel?

Q3: Is the fireplace currently in use?

Figure 7. Examples from IDKVQA, showing images and the questions generated by the LLM.

the trade-off between risk and coverage in a VQA model by assigning a reward to questions that are answered correctly, a penalty c to questions answered incorrectly, and a zero reward to the model abstaining. Formally:

$$\Phi_c(x) = \begin{cases} \text{Acc}(x), & g(x) = 1 \text{ and } \text{Acc}(x) > 0 \\ -c & g(x) = 1 \text{ and } \text{Acc}(x) = 0 \\ 0, & g(x) = 0 \end{cases}$$

Here, $x = (i, q) \in \mathcal{X}$ is the input pair where i is the image and q is the question. The function $g(x)$ is equal to 1 if the model is answering and 0 if it abstains. The parameter c denotes the cost for an incorrect answer, and the VQA accuracy Acc is:

$$\text{Acc}(f(x, y)) = \min\left(\frac{\text{\# annotations that match } f(x)}{3}, 1\right)$$

where the function $f : \mathcal{X} \rightarrow V$ output a response $r \in \mathcal{R}$ for each input pair x.

Baselines. We evaluate our proposed *Normalized Entropy* against three baseline methods:

(i) MaxProb, which selects the response r with the highest predicted probability from the VLM, given image i and question q. Formally, $r = \text{VLM}(i, q)$. It does not incorporate uncertainty estimation.

(ii) LP [70], a recently proposed Logistic Regression model trained as a linear probe on the logits distribution of the first generated token. The model is trained on the *Answerable/Unanswerable* classification task using the VizWiz VQA dataset [15], which includes $23,954$ images for training. When applied to IDKVQA, the logistic regression model first predicts whether the question q is *Answerable* or *Unanswerable*. If the question is deemed answerable, the response r with the highest probability is selected among $\{$`Yes, No`$\}$; otherwise, the response `I don't know` is returned.

(iii) Energy score, an energy-based framework for out-of-distribution (OOD) detection [29]. Following the implementation in [29], an energy score is computed to identify whether the given question-image pair is OOD. If the pair is classified as OOD, the response `I don't know` is returned; otherwise, the response with the highest probability is selected among $\{$`Yes, No`$\}$.

Finally, for our proposed *Normalized Entropy* estimation, we link the abstention function $g(x)$ (*i.e.*, determining whether the model abstains from answering) to Eq. 3 in the main paper. Specifically, $g(x) = 1$ if the Normalized Entropy classifies the model as *certain*, and $g(x) = 0$ otherwise. Then, if the model is deemed certain, we return the most probable answer $\{$`Yes, No`$\}$; otherwise, the response `I don't know` is selected.

H.3. Sensitivity analysis of the threshold τ

This section provides additional details about how small variations of the threshold parameter τ affect both our Normalized Entropy technique (Eq. 3) and the Energy Score [29], with respect to the target metric $\Phi_{\text{c=1}}$.

To conduct this analysis, we perform an ablation study on datasets of varying sizes, obtained by randomly subsampling CoIN-Bench. Specifically, we create five sets containing 50% of the question-answer pairs from CoIN-Bench, five sets comprising 70% of the question-answer pairs, and also use the full dataset (100%) for a total of 11 datasets.

For each dataset, we identify the optimal threshold τ^* for each method through an exhaustive search over predefined ranges, resulting in 22 optimal thresholds (11 per method)

Around each τ^*, we define a neighborhood $\boldsymbol{\tau}$ comprising 30 new thresholds τ sampled symmetrically around it. Our goal is to analyze how $\Phi_{\text{c=1}}$ changes across these neighborhoods: if the values are spread out, it means that the method is very sensitive to small changes of τ near the optimal value, whereas if they are more tightly distributed it means that it is more robust.

Therefore, for each method and related neighborhood $\boldsymbol{\tau}$, we compute 30 Φ_c values, one for each $\tau \in \boldsymbol{\tau}$, and *normalize* them to the range$[0,1]$ by dividing each value by the best $\Phi_{c=1}$ found in $\boldsymbol{\tau}$. We do so to measure only the distribution of the $\Phi_{c=1}$ values, not their absolute values, and to help the comparison across datasets of the same size (otherwise, due to chance, they could have distributions of different values). Finally, we aggregate all these normalized $\Phi_{c=1}$ scores across dataset size, resulting in Fig.3 (main paper).

From the figure, we can see that our technique has smaller interquartile ranges and tighter distributions of $\Phi_{c=1}$, while the Energy Score [29] exhibits larger tails, indicating more variance. Moreover, our method shows distributions more biased toward higher values (which would indicate smaller degradation *w.r.t.* the best $\Phi_{c=1}$) than those of the Energy Score, and this gap increases as the dataset size decreases. This shows that our technique is generally more robust, especially in data-scarce situations, and less sensitive to small variations in τ.

I. Prompts

I.1. P_{init} - Initial Description

```
P_init = """Describe the {target_object} in the
    provided image."""
```

I.2. $P_{details}$ - Gather Additional Information

```
P_details = """You are an intelligent embodied
    agent equipped with an RGB sensor, an object
    detector, and a Visual Question Answering (
    VQA) model.
Your task is to explore an indoor environment to
    find a specific target {target_object}.
The detector has identified a {target_object}.
    The VQA model has provided the following
    description of the scene:

<START_OF_DESCRIPTION>
{distractor_object_description}
<END_OF_DESCRIPTION>

Based on your past interactions with the user,
    you know the following facts about the target
     picture:
<START_TARGET_PICTURE_FACTS>
{facts_about_the_target_picture}
<END_TARGET_PICTURE_FACTS>

Your task is to:
- ask more question to the VQA model on the
    detected {target_object} to maximize
    information gain.

Ensure your output follows the following format:

YAML_START # must be present to get the
    information back
attributes_of_the_image:
    <attribute name>: "<attribute value>" #
    summarize all the known attributes from the
    description, enclosed in " "
questions:
        <question_number>: "<question content>"
YAML_END # must be present to get the information
     back

Provide your reasoning step-by-step, after the
    YAML_END tag."""
```

I.3. P_{check} - Check detection with LVML

```
 P_check = """Does the image contain a {
    target_object}? Answer with Yes, No or ?=I
    don't know."""
```

I.4. $P_{selfquestion}$ - Extract attributes and generate Self-Questions

```
P_ATTRIBUTES_AND_SELF_QUESTIONS = """
You are an intelligent embodied agent equipped
    with an RGB sensor, an object detector, and a
     Visual Question Answering (VQA) model. Your
    task is to explore an indoor environment to
    find a specific target {target_object}.
The detector has identified a {target_object}.
    The VQA model has provided the following
    description of the scene:

<START_OF_DESCRIPTION>
{distractor_object_description}
<END_OF_DESCRIPTION>

Based on your past interactions with the user,
    you know the following facts about the target
     picture: <START_TARGET_PICTURE_FACTS> {
    facts_about_the_target_picture} <
    END_TARGET_PICTURE_FACTS>

Assume that the detected image description
    contains hallucinations. Your goal is to
    verify every attribute of the detected {
    target_object} description through questions.
     Formally:
- Detect possible hallucinations in the VQA model
    's description
- Get more information about the detected object.
Every question should be in this format: "<
    question content>? You must answer only with
    Yes, No, or ?=I don't know." This allows us
    to assess the likelihood of the answers.


Ensure your output follows the following format:
YAML_START # must be present to get the
    information back
attributes_of_the_image:
    <attribute name>: "<attribute value>" #
    summarize all the known attributes from the
    description, enclosed in " "

questions_for_detected_object: # question for the
     detected object, if any
    <Question number>:  "<question>? You must
    answer only with Yes, No, or ?=I don't know."
reasoning_for_detected_object:
    <Question number>: <reasoning>
YAML_END # must be present to get the information
     back
```

```

Provide your reasoning step-by-step, after the
    YAML_END tag."""
```

I.5. $P_{refined}$ - Refined image description

```
P_refined = """
You are an intelligent embodied agent equipped
    with an RGB sensor, an object detector, and a
     Visual Question Answering (VQA) model.
Your task is to refine an image description based
     on certainty estimates and user interactions
    .

Scenario:
The detector has identified a scene with a {
    target_object}. The VQA model provided this
    initial scene description:

<START_OF_DESCRIPTION>
{distractor_object_description}
<END_OF_DESCRIPTION>


Questions asked and responses:
<START_QUESTION_AND_RESPONSES>
{list_questions_answers_uncertainty_labels}
<END_QUESTION_AND_RESPONSES>

Task:
Using the questions/answer pairs with uncertainty
     labels, refine the image description.
Since we have to find a {target_object}, put
    enphasis on it. Do not include in the
    description information that is labeled as
    uncertain.

Ensure your response follows the format below:
YAML_START # must be present to get the
    information back
attributes_of_the_image:
    <attribute name>: "<attribute value>" #
    summarize all the known attributes from the
    description, enclosed in " "
image_description_refined: <insert refined
    description>  # Ensure that the string does
    not contain a newline (\n) after the tag
    image_description_refined:
YAML_END # must be present to get the information
     back

Provide your reasoning step-by-step, after the
    YAML_END tag."""
```

I.6. P_{score} - Alignment score

```
P_score = """
You are an intelligent agent equipped with an RGB
     sensor, object detector, and Visual Question
     Answering (VQA) model.
Your goal is to identify a target {target_object}
     based on a scene description and prior
    knowledge of the target.

Scenario:
The object detector has identified a scene
    containing a {target_object}, and the VQA
    model has provided the following description:

<START_OF_DESCRIPTION>
{distractor_object_description}
<END_OF_DESCRIPTION>

Target object information:
Based on previous interactions, you know the
    target picture has the following
    characteristics:
<START_TARGET_PICTURE_FACTS>
{facts_about_the_target_picture}
<END_TARGET_PICTURE_FACTS>

Task:
1. Similarity analysis.
Analyze how closely the detected scene
    description aligns with the known facts about
     the target {target_object}. Provide a
    similarity score between 0 and 10, where:
- 0 = The detected {target_object} is not the
    target object.
- 10 = The detected {target_object} is definitely
     the target object.
- If no information about the target is available
    , the score should be -1.

2. Question Generation:
- The question is for the target object, not the
    detected one.
- Ask exactly one specific, relevant, and human-
    answerable question related to the target
    object that maximizes information gain for
    identifying the target {target_object}.
- Do not ask speculative or irrelevant questions
- The question should be grounded in observable
    or known details from the scene, focusing on
    key characteristics that can help confirm or
    refute the identity of the target object.

Ensure your response follows the format below:
YAML_START # must be present to get the
    information back
similarity_score: <similarity score>
questions:
    <question_number>: <question_content>
YAML_END # must be present to get the information
     back

Provide your reasoning step-by-step for the
    similarity score and questions, after the
    YAML_END tag."""
```

J. Algorithm

We first present the complete AIUTA's algorithm in Sec. J.1. As outlined in the main paper (Sec. 4), AIUTA enriches the zero-shot training policy VLFM [63]. Specifically, we detail the input/output structure of AIUTA regarding VLFM policy π, as well as AIUTA's main component, *i.e.*, the *Self Questioner* (see Sec. J.2) and the *Interaction Trigger* (Sec. J.3).

J.1. AIUTA Algorithm

Algorithm 1 outlines the complete AIUTA pipeline. Upon detecting a candidate object, AIUTA first invokes the *Self*

Questioner module (shown in Sec. J.2) to obtain an accurate and detailed understanding of the observed object and to reduce inaccuracies and hallucinations, obtaining a refined observation description $S_{refined}$. Then, with the known facts about the target instance and the refined description, AIUTA invokes the *Interaction Trigger* module (Sec. J.3) for up to 4 iterations rounds (*i.e.*, Max_Iteration_Number $= 4$), as specified in Sec. 6 under **Implementation Details**. Within each interaction round, if AIUTA returns the `STOP` action, then the policy π terminates the navigation since the target instance is found; otherwise, the policy π continues the navigation process.

Algorithm 1 AIUTA

```
Require: Target object facts $F$, Observation $O_t$, policy $\pi$, Candidate Object Detection, Max Iteration number
                                          ▷ Upon candidate object detection
1: $S_{refined} \leftarrow$ Self_Questioner$(F, O_t)$ ▷ enrich details and reduce inaccuracy, obtain a refined description
2: if $S_{refined} =$ " " then
3:     $\pi$(CONTINUE_EXPLORING) ▷ VQA detection failed, Signal to policy $\pi$ to continue exploration
4: for each iteration in Max_Iteration_Number do
5:     aiuta_action $\leftarrow$ Interaction_Trigger$(F, S_{refined})$
6:     if aiuta_action = STOP then
7:         $\pi$(STOP) ▷ Signal to policy $\pi$ that the object is found! Terminate exploration
8:     else
9:         $\pi$(CONTINUE_EXPLORING) ▷ Signal to policy $\pi$ to continue exploration
```

J.2. Self Questioner

Algorithm 2 Self Questioner Module

```
Require: Target object facts $F$, Uncertainty Threshold $\tau$, Observation $O_t$, $P_{init}, P_{details}, P_{check}, P_{selfquestions}, P_{refined}$
1: Step 1: Detailed Detection Description, from $S_{\text{init}}$ to $S_{\text{enriched}}$
2: Initial scene description: $S_{\text{init}} \leftarrow \text{VLM}(O_t, P_{\text{init}})$
3: Self-generate questions to enrich description
       $Q^{details}_{a \to a} \leftarrow \text{LLM}(P_{\text{details}}, S_{\text{init}}, F)$
4: for each question $q_j$ in $Q^{details}_{a \to a}$ do
5:     $r_{a \to a} \leftarrow \text{VLM}(O_t, q_j)$                ▷ Get answers
6:     $S_{\text{init}} \leftarrow \text{concatenate}(S_{\text{init}}, r_{a \to a})$
7: $S_{\text{enriched}} \leftarrow S_{\text{init}}$              ▷ Updated scene description
8: Step 2: Perception Uncertainty Estimation
9: $(r_{\text{check}}, u_{\text{check}}) \leftarrow \text{VLM}(O_t, P_{\text{check}})$ ▷ Check detection with uncertainty
10: if NOT ($r_{\text{check}} =$ "Yes" AND $u_{\text{check}} =$ "Certain") then
11:     return " "   ▷ empty string, thus continue exploring
12: $Q^{attribute}_{a \to a} \leftarrow \text{LLM}(P_{\text{self questions}}, F, S_{\text{enriched}})$ ▷ Generate self-questions to verify attributes
13: Container $\leftarrow \{\}$  ▷ Store question, answer, uncertainty
14: for each question $q_j$ in $Q^{attribute}_{a \to a}$ do
15:     $(r_j, u_j) \leftarrow \text{VLM}(O_t, q_j)$ ▷ Get answers and uncertainties
16:     Container $\leftarrow$ concatenate(Container, $\{q_j, r_j, u_j\}$)
17: Step 3: Detection Description Refinement
18: $S_{\text{refined}} \leftarrow \text{LLM}(P_{\text{refined}}, \text{Container}, S_{\text{enriched}})$ ▷ Filter out uncertain attributes
19: return $S_{\text{refined}}$
```

J.3. Interaction Trigger

Algorithm 3 Interaction Trigger

```
Require: Target object facts $F$, Refined observation description $S_{\text{refined}}$, $P_{score}$, $\tau_{stop}$ and $\tau_{skip}$
1: $(s, q_{a \to u}) \leftarrow \text{LLM}(P_{score}, S_{\text{refined}}, F)$ ▷ get alignment score $s$, and question for the human $q_{a \to u}$
2: if $s \geq \tau_{stop}$ then
3:     return STOP    ▷ target found, stop navigation.
4: else if $s < \tau_{skip}$ then
5:     return CONTINUE_EXPLORING ▷ skip the question and continue exploring
6: else
7:     $r_{u \to a} \leftarrow \text{Ask\_Human}(q_{a \to u})$ ▷ posing clarifying question $q_{a \to u}$ from the agent to the human.
8:     $F \leftarrow \text{Update\_Facts}(F, r_{u \to a})$ ▷ update target object facts $F$
```