Article | Received 19 March 2025; Accepted 23 June 2025; Published 9 July 2025
https://doi.org/10.55092/rl20250004

Utility of low-parameter count, quantized large language models and visual language models in semantic navigation †

Arturs Elksnis[1], Ziheng Xue[1], Martin Pearson[1] and Ning Wang[2,*]

[1] Bristol Robotics Laboratory, University of the West of England and University of Bristol, Bristol, UK

[2] School of Computing and Digital Technologies, Sheffield Hallam University, Sheffield, UK

† This work is an extended version of the paper presented at the 2024 29th International Conference on Automation and Computing (ICAC), and copyright permission has been obtained from IEEE for its publication in this journal.

* Correspondence author; E-mail: ning.wang@shu.ac.uk.

Highlights:

- Use of Visual Language Model in Extraction of Semantic Information.
- Use of Large Language Model in Extraction of Semantic Information.
- Semantic Navigation with Local Models.

Abstract: The use of Large Language Models (LLMs) in Semantic Navigation (SN) is a novel approach and has so far centered around high-parameter count cloud-hosted LLMs. In this study we explore the capabilities of low parameter count (8.54×10^9 or less) quantized local LLMs that can be hosted on a consumer grade laptop. We demonstrate the use of such LLMs in extracting semantic knowledge from the environment which is then used, in a simulated environment, to create an executable path to lead the agent to a goal set by a human operator. Toward this, we explore two main tasks for which LLM can be employed: room classification and goal selection, and compare the performance of several LLMs and an appropriately constructed Support Vector Classifier (SVC) for this purpose. We then present a simple framework for evaluating Visual Language Models (VLMs) and LLMs in SN. We compare 4 LLMs-Gemma, Llama3 and two quantizations of Mistral. Among the LLMs we tested, Llama3 is the most accurate room classifier whereas goal selection performance varies between rooms and different LLMs. Finally, we evaluated 3 Visual Language Models (VLMs)- 4-bit quantized form of Chameleon, un-quantized Chameleon and un-quantized Moondream2. We found that they are best used for hard-to-classify scenes where a list of objects cannot be extracted for classification with an LLM. Our combined approach takes advantage of the best aspects of both types of model. The source code and test data for this project are available at: https://github.com/archie1983/cvm_semantic_navigation.

Keywords: semantic navigation; semantic reasoning; large language model; quantized LLM; visual



Copyright©2025 by the authors. Published by ELSP. This work is licensed under a Creative Commons Attribution 4.0 International License, which permits unrestricted use, distribution, and reproduction in any medium provided the original work is properly cited.

language model; Gemma; LLama3; Mistral; Chameleon; Moondream2

1. Introduction

Semantic Navigation (SN) is a concept where an agent navigates a 2D or 3D space based on semantic understanding of that space; e.g., given a target and the contents of a room, the agent should be able to autonomously infer the likely location of the target within the room (even if not observed directly) and navigate there, e.g., [1–3]. This concept can be extended to multi-room habitats as shown in [4–6]. These concepts can improve the autonomy of an agent leading to a system that can autonomously navigate to a target which has been set in natural human language, e.g., "go to the pantry and bring me a chocolate bar".

Autonomous agents that can interact with humans and the environment in this way could find a wide range of uses in areas like healthcare, hospitality, service jobs, elderly care, *etc.*, which sets strong motivation to research SN.

SN uses the semantic relationships that exist between objects and locations in the real world. Humans use such relationships naturally, e.g. if the desired object was a bar of chocolate, then the natural locations to look for it in a human habitat would include places like pantry, snack box, drawer, *etc.* In addition, the natural room for searching for the pantry or snack box might be a kitchen. For machines, however, the available technical options and therefore potential implementations of SN are constantly changing and evolving.

Historically SN used to rely on ontologies, e.g., [7], to build knowledge-linked semantic object maps using pre-existing ontologies defined in Web Ontology Language (OWL) and create such ontologies from free text knowledge bases such as OMICS [8]. Other notable SN works that rely on ontologies include [9] and [10].

A more recent approach used Graph Convolutional Networks (GCN) to encode priors, essentially storing semantic knowledge that way [2]. This GCN is then used to infer potential locations for required objects and generate navigation paths using the semantic knowledge encoded earlier.

Other methods useful for capturing semantic information include ConceptNet graph of general knowledge [11], embedding methods like GloVe [12] and parts of transformer architecture [13] like encoders, decoders and attention mechanism. Interestingly, all of the above are used in [14] leading to remarkable improvements in solving REVERIE [15] tasks. For context, REVERIE is a navigation task that is set in high-level natural language and requires localization of a remote target object that cannot be immediately observed. The following is an example from the REVERIE task dataset: *"There is a bottle in the office alcove next to the piano. It is on the shelf above the sink on the extreme right. Please bring it here."*.

The recent advent of Large Language Models (LLMs) offers new opportunities in the SN area. LLM training data often include rich semantic information, and LLMs themselves are based on transformer architecture, components of which are clearly fundamental in [14].

We were curious about the utility of LLMs in SN, but soon realized that most prior works on LLM based SN typically use high parameter count cloud-hosted LLMs [16–20]. Cloud-hosting presents several disadvantages, e.g., lack of control over the model, requirement to be connected to the internet,

subscription cost and data privacy issues, especially in domestic and care environments. SN capable agents can potentially build accurate maps of user's habitat and accumulate detailed knowledge of locations of various objects, including sensitive items, e.g. money, documents, jewelry, *etc.* Leaking such information outside of the user's premises would be an unacceptable privacy risk and is an obvious problem with cloud-hosted LLMs. LLMs can be run locally, but typically they have high memory requirements, e.g., running inference with full Llama2-70B requires nearly 150GB of memory, which is typically video memory (VRAM) [21].

We were interested in the suitability of LLM for SN on relatively resource-constrained platforms. We therefore focused on modest-size local quantized LLMs to demonstrate aspects of SN. LLM quantization is a process in which the weights and activations of a neural network are reduced to take up less space. This shrinks the size of the model and makes inference faster at the cost of accuracy.

An equation for calculating memory requirements of LLM is given in [22]:

$$M = \frac{P * 4}{32/Q} * 1.2 \tag{1}$$

where M is the amount of memory required for inference, P is the number of parameters of the LLM architecture, 4 represents the 4-byte size of an unquantized parameter, Q represents the number of bits actually used per parameter (e.g. $Q = 6$ when 6-bit quantization is used), 32 is the number of bits in a 4-byte weight and 1.2 is an overhead factor. Although the equation is simplistic, we can use it to gain insights into LLM memory requirements by parameter count and quantization factor. Figure 1 illustrates this relationship; e.g., if available memory is $12GB$, then, for realistic use, choice of LLM is constrained to 7 or 8 billion parameters and 4 or 6 bit quantization. Inference time is also a factor, in general it follows a similar trend as the memory requirements, e.g., Table 4 reveals inference time differences between 4-bit and 6-bit Mistral LLM. We found that quantization also reduces the model's conformance to the given instructions, which introduces challenges in interpreting the model's output. For example, we noticed that quantized low-parameter-count LLMs tend to ignore the instruction to reply with a single word when inferring the room type. Likewise, the instruction to conform to a specific sentence structure to simplify interpretation tends to be ignored. This and other non-conformities make some LLMs better choices than others, selection of which is at the core of our research in this work.

We explore several pre-trained LLMs that are freely available: Llama3, Gemma and 2 quantizations of Mistral, which we do not modify. We also explore 3 Vision Language Models (VLM): Moondream2 and 2 quantizations of Chameleon, which, we find, can complement the LLM results.

For clarity, we don't attempt solving a navigation task like REVERIE or present a full SN solution. Building from initial prior work in this area [23] our main contribution is the evaluation of low-parameter count quantized LLM performance in two key tasks in SN: **room classification** and **goal selection**. These tasks are important components of SN in the domain of domestic assistance [6, 18, 24, 25]. Further, we present an evaluation of low-parameter count VLM performance in room classification and a method of combining LLM and VLM outputs for a significantly improved complementary result.

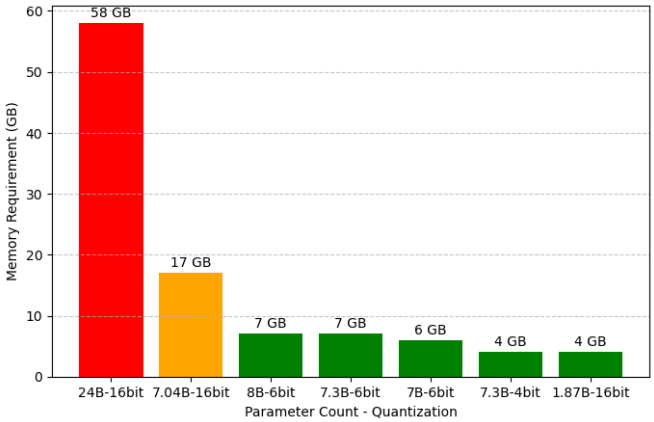


Figure 1. LLM memory requirements by parameter count and quantization factor. Green bars represent model memory footprints that we could run on our laptop (12GB VRAM). Orange bar represents the model that we could only run on our JETSON Xavier platform (32GB VRAM). Red bar is an example of model that was not available to use due to its memory requirements.

The paper is organized as follows: An overview of related work of LLMs in SN is given in Section 2. Our experiments are explained in Section 3. followed by testing results and comparison of LLM, SVC and VLM in Section 4. Finally, we discuss these results and highlight pathways for future research in Section 5.

2. Related work

Huang *et al.* propose a framework in which an LLM is used to integrate low-level, short-horizon robotic skills together with continuous feedback from the environment that lead to successful achievement of high-level tasks [26]. This work was later extended to use an LLM for successfully breaking down high-level tasks into smaller, actionable steps that can be executed by an agent [20]. This is an interesting concept, however, it episodically suffers from the LLM being too unrestricted leading to output actions that are not in the required format, are compound, or ambiguous, requiring the adoption of cosine similarity function to resolve. We instead try to explicitly limit the choices that an LLM is allowed to draw from.

A different approach involved first generating a map with embeddings of detected landmarks from previously captured RGB-D images [19]. An LLM was then used to extract goals from free-text human language which were then matched to the embeddings on the map. Finally, LLM code writing capability is used to generate Python code that is directly executed by the robot to arrive at the set goal using the generated map. Whilst impressive, their solution relies on the object (e.g. laptop) having been first seen and mapped instead of using LLM to infer where the object is likely to be.

To address this, a navigation method that uses LLM to extract heuristics about where to find the required object in an unknown scene has been proposed [18]. For example, if the agent was tasked to find a toilet but can currently only observe a kitchen and living room, then we can ask an LLM where is it more likely to find the toilet next to the kitchen or next to the living room. An appropriate LLM would reply that the toilet can be found in the bathroom and that the bathroom is more likely to be next to the living room and not the kitchen. That reply is then used as a heuristic in a Frontier-Based-Exploration (FBE) algorithm to navigate and first explore the living room to eventually find the bathroom and toilet. If the bathroom is not found while exploring the living room, then FBE leads next to exploration of the less

likely kitchen for the goal. It has been demonstrated that this approach reduces the time it takes to find a target compared to vanilla FBE. They have also employed several interesting techniques of using LLMs providing significant inspiration for our work and like us they explicitly rely on room classification and goal selection sub-tasks. Their approach however explores the environment while looking for the target and therefore does not have a complete semantic knowledge about the setting- at least not at the start. While this approach certainly has its merits, and its core use-case being exploring unknown environments, it would be less practical for fixed environments, e.g., hospital or home setting, where an exploration phase can precede the operation phase. Our approach first builds semantic knowledge during the exploration phase and then uses this knowledge to find requested objects during the operation phase relying on the decisions made by the LLM.

Further inspiration was taken from Qiao *et al.*, who demonstrated the use of a low-parameter-count LLM for what they call a co-pilot role for their SN agent [27]. They accumulate navigation data along the way (e.g. location history and previous observations) and then use the gathered data to query their LLM when the agent's confusion score reaches a pre-defined threshold. It is essentially a two step process- they first query LLM for a type of clarification that is best suited at a given location and then request the actual clarification of the required type. This leads to impressive results on REVERIE tasks.

Long *et al.* introduce an interesting use of various LLMs and VLMs as different experts to solve navigation tasks [17]. They define five sub-tasks: Instruction Analysis, Vision Perception, Completion Estimation, Movement Decision and Decision Testing. These sub-tasks are then solved by a collection of appropriately prompted LLMs and VLMs acting as experts for each of the sub-tasks. This was then used to attempt to unify different types of navigation tasks: goal navigation, visual language navigation, and demand driven navigation using an LLM to interpret the incoming task and generate actions at each step [16].

Finally, there has been some evaluation of several open-source LLMs that aligns with our objectives [28], although focused on higher parameter count LLMs than in this study and evaluated against different tasks.

In our experiments, we use LLMs to evaluate their performance in two key tasks in SN: **room classification** and **goal selection**. We also explore the potential of using VLM in room classification and make a case for a hybrid approach of using both LLM and VLM. We then quantitatively evaluate the accuracy of room classification against ground truth. Finally, we perform both qualitative and quantitative analysis of goal selection success.

3. Implementation & experiments

The two SN tasks and scenarios that we used to evaluate local LLMs are as follows:

Room classification

- The setting is a human habitat (e.g. a flat) on a single level with 4–10 rooms (each functionally labelled as being either bathroom, kitchen, bedroom or living room) from ProcTHOR-10k [29] dataset. We only used habitats with all 4 room types present to allow for the richest sets of discoverable objects.
- The agent explores the habitat and builds semantic knowledge by classifying the rooms and locating them in a map.

Goal selection

- Human operator defines an item that needs to be retrieved in free text (e.g. "a fresh, cold bottle of beer") which has not necessarily been observed by the agent in the exploration phase.
- The agent has to first determine the appropriate room (e.g. kitchen) and then within the room- the location that will most likely contain the requested item (e.g. fridge) as described in Section 3.4.

We evaluate 4 instruction-tuned LLMs with weights quantized to 6 and 4 bits as shown in Table 1: Gemma [30], Mistral [31] and LLama3 [32] and a Support Vector Classifier (SVC) for comparison.

Table 1. LLMs evaluated in this study.

Model	Parameter	Quantization	Memory	Ollama Tag	URL
Gemma	8.54×10^9	6 bit	7.16GB	gemma:7b-instruct-v1.1-q6_K	https://ollama.com/library/gemma:7b-instruct-v1.1-q6_K
Mistral	7.24×10^9	4 bit	4.05GB	mistral:7b-instruct-v0.2-q4_0	https://ollama.com/library/mistral:7b-instruct-v0.2-q4_0
Mistral	7.24×10^9	6 bit	6.07GB	mistral:7b-instruct-v0.2-q6_K	https://ollama.com/library/mistral:7b-instruct-v0.2-q6_K
Llama 3	8.03×10^9	6 bit	6.74GB	llama3:8b-instruct-q6_K	https://ollama.com/library/llama3:8b-instruct-q6_K

The SVC was trained using training data hybridization techniques as described in [24]. First, we associated all objects to the room types that they may be observed in using our own understanding of the world. We then picked a random room type and generated a random set of objects for the room from the previously defined association. We generated 1000 such training samples for the SVC. The original AI2-THOR [33] dataset was used for the detectable objects which (although visually similar) differ from the objects in ProcTHOR-10k scenes. This introduces some extra uncertainty for the SVC as it might get queried on tokens it has not been trained on thus bringing the scenario closer to real life. All these experiments were conducted using a consumer-grade laptop computer (Intel i7-11700H @2.3GHz, 32GB RAM, Nvidia GeForce 3060 with 12GB VRAM, Ubuntu 20.04.6.)

We then repeated the room classification experiment using three candidate VLMs: Moondream2 [34], Chameleon [35] and a 4-bit quantized form of Chameleon (see table 2 for details). Our criteria for choosing these VLMs was the same as the LLMs- we want to run them locally on easily accessible hardware. The laptop used for the initial LLM experiments was replaced by an NVIDIA Jetson Xavier platform [36] Single Board Computer (SBC). The 4.5GB memory footprint of Moondream2 would have allowed us to use the laptop as used for LLM experiments, however, the 17GB memory footprint of Chameleon in 16-bit, un-quantized form would have been too large. We therefore also evaluated a 4-bit quantized version of Chameleon with a 4.2GB memory footprint. The un-quantized form of Chameleon however, still meets our selection criteria for VLM since we identified that the largest VRAM available for a consumer grade laptop computer at the time of writing was 24GB.

Table 2. VLMs evaluated in this study.

Model	Parameter Count	Quantization	Memory (GB)	Download Link
Chameleon	7.04×10^9	16 bit	15.74	https://huggingface.co/facebook/chameleon-7b
Chameleon	7.04×10^9	4 bit	3.94	https://huggingface.co/facebook/chameleon-7b
Moondream2	1.87×10^9	16 bit	4.18	https://huggingface.co/vikhyatk/moondream2

We use AI2-THOR v5.0.0 framework and Thortils [37] v3.3.4 package to explore habitats which we get from the ProcTHOR-10k [29] dataset.

3.1. Model quantization

All quantized versions of the LLMs, that we used, were provided by the original authors of the models. The download sources for the models are given in Tables 1 and 2. The quantization was performed using GGUF [38] method with further details available in the respective model cards.

The 4-bit quantized version of Chameleon VLM, that we used, was quantized using 4-bit NormalFloat (NF4) quantization via the BitsAndBytes library [39], with double quantization. All quantized operations were computed in *torch.float16* precision. Please see Appendix 1 for details.

3.2. Prompting considerations

Parsing and interpretation of the models' output is significantly simpler when it is short, concise and structured. From experience, eliciting such outputs can be challenging with lower parameter-count models. Prompting techniques significantly affect the output of LLM and VLM models, for example, the Chain-of-Thought (CoT) prompting technique [40] can be used to induce more in-depth reasoning. Structure and layout of the prompt is also important, we found that the placement of new-line characters and whitespace as shown in Figure 2 affects the quality of the output.

3.2.1. LLM prompting considerations

In total, we use four prompts, all of which are shown in Figure 2. Figure 2a was used for the room classification experiments, while prompts Figure 2c and Figure 2b were used for goal selection (room and object proximity respectively). Prompt Figure 2d was used for additional experiments descibed below. As in related studies, we also adopted a Chain-of-Thought (CoT) prompting technique by requiring the LLM to justify its answers[18]. This worked reliably, however, formatting differences between answers have been observed. We also noted that the intended room type was occasionally incorporated into the reasoning part of the answer and not stated separately on the first line (more frequently from Llama3 and Mistral). Other room types would also be mentioned in the reasoning part although the intended room type was predominantly mentioned earlier than other types. These inconsistencies present a challenge in interpreting the answer which ultimately impacts on the accuracy of the interpreted result. To mitigate this, we parsed the full answer from the LLMs and chose the earliest mention of any of the room types as its answer. This approach does not guarantee the extracted answer to be interpreted correctly as the LLM might first state the least likely room type to rule it out and then proceed with its reasoning about the better candidates (akin to human conversation). From empirical observations this mostly happened with High Confusion Level (HCL) object sets as explained in Section 3.3.1 and discussed in Section 4. The structured querying strategies, as suggested in [18], lead to more uniform answers that were easier to analyze programmatically. However, we observed that their use occasionally confuses Gemma LLM, sometimes causing the inference process to terminate abnormally. We attributed this to deficiencies that will be solved in future releases of Ollama [41] and/or Gemma and, therefore, chose not to use

structured querying strategies in favor of a framework that allows comparing all LLMs with the same prompts. Similarly, negative prompting did not lead to significantly better results, which we attributed to our prompts being simpler to those used in other works [18]. Finally, we experimented using prompt Figure 2d which requests the LLMs to rank the given objects in the list when choosing where to look for the required item. This consistently gave the same first answer as the prompt Figure 2b, but also provided a reasonable list of other options to explore, which would be useful for real-life scenarios.

```
I observe the following objects while exploring a room:
{0}

What kind of room is this?
1. Living room
2. Kitchen
3. Bedroom
4. Bathroom

You should always provide justification
```

(a) Room classification prompt.

```
I am looking for:
{0}

Which object is it most likely to be near?

{1}

You should always provide justification
```

(b) Goal selection prompt.

```
I am looking for:
{0}

Which room should I look in first?

1. Living room
2. Kitchen
3. Bedroom
4. Bathroom

You should always provide justification
```

(c) Room selection prompt.

```
I am looking for:
{0}

Which object is it most likely to be near?

{1}

Please give rank for each object.
You should always provide justification
```

(d) Room selection prompt - ranked.

Figure 2. Prompts used for querying LLM. **(a)**: "{0}" - placeholder for a list of objects; **(b) & (d)**: "{0}" - placeholder for user's goal, "{1}" - placeholder for a list of objects observed in a given room; **(c)**: "{0} - placeholder for user's goal".

3.2.2. VLM prompting considerations

The models that we use respond differently to CoT techniques, specifically, Moondream2 tended to ignore requests to restrict its choice of room categories—it often didn't limit its response to 4 categories of room. To balance this, we also evaluated prompts where we extended the permissible categories to 6 by including Office and Storage. We did so with all models.

There are also different options for passing the image to the models, Chameleon requires a placeholder of the image within the text part of the prompt, while Moondream2 does not. From the performance of 4-bit quantized Chameleon, placing the image placeholder right after referencing the image in prompt, works markedly better than placing it at the end of the prompt. Finally, asking the VLM to provide reasoning for the answer affects the result in different ways depending on the choices described above.

This led to 6 different prompts that we evaluated for Moondream2 and further 11 prompts that we evaluated for two quantizations of Chameleon VLMs.

The prompts that we used for VLMs are listed in Table 3.

Table 3. Prompts used for VLMs. CoT denotes whether the VLM was prompted to provide reasoning for its answer. Image location describes the location of image placeholder within the prompt. Prompt ID references the prompt to other parts in the text.

Prompt	CoT	Label Count	Image Location	Prompt ID
What kind of room is in this image? Please provide reasoning for your answer.	yes	0	NA	P1
What kind of room is in this image? Please provide reasoning for your answer. You may choose one from the following categories: kitchen, bedroom, bathroom, living room.	yes	4	NA	P2
What kind of room is in this image? Please provide reasoning for your answer. You may choose one from the following categories: kitchen, office, bedroom, bathroom, living room, storage.	yes	6	NA	P3
What kind of room is in this image?	no	0	NA	P4
What kind of room is in this image? You may choose one from the following categories: kitchen, bedroom, bathroom, living room.	no	4	NA	P5
What kind of room is in this image? You may choose one from the following categories: kitchen, office, bedroom, bathroom, living room, storage.	no	6	NA	P6
What kind of room is in this image?<image> Please provide reasoning for your answer.	yes	0	middle	P7
What kind of room is in this image?<image> Please provide reasoning for your answer. You may choose one from the following categories: kitchen, bedroom, bathroom, living room.	yes	4	middle	P8
What kind of room is in this image?<image> Please provide reasoning for your answer. You may choose one from the following categories: kitchen, office, bedroom, bathroom, living room, storage.	yes	6	middle	P9
What kind of room is in this image?<image> You may choose one from the following categories: kitchen, bedroom, bathroom, living room.	no	4	middle	P10
What kind of room is in this image?<image> You may choose one from the following categories: kitchen, office, bedroom, bathroom, living room, storage.	no	6	middle	P11
What kind of room is in this image? Please provide reasoning for your answer.<image>	yes	0	end	P12
What kind of room is in this image? Please provide reasoning for your answer. You may choose one from the following categories: kitchen, bedroom, bathroom, living room.<image>	yes	4	end	P13
What kind of room is in this image? Please provide reasoning for your answer. You may choose one from the following categories: kitchen, office, bedroom, bathroom, living room, storage.<image>	yes	6	end	P14
What kind of room is in this image?<image>	no	0	end	P15
What kind of room is in this image? You may choose one from the following categories: kitchen, bedroom, bathroom, living room.<image>	no	4	end	P16
What kind of room is in this image? You may choose one from the following categories: kitchen, office, bedroom, bathroom, living room, storage.<image>	no	6	end	P17

3.3. Room classification

Prior to classifying rooms based on their content, a pseudo-random data set was collected using a Habitat Exploration Algorithm1 as illustrated in Figure 3. After loading the habitat in the simulation, we select a set of random locations $L = \{l_i\}_{i=1}^{N}$ with N depending on the size of the habitat. We teleport to each of the locations in set L and capture the First-Person View (FPV) from the agent taken at a randomly selected number of yaw rotations sampled from the set $PYR = \{0, 45, 90, 135, 180, 225, 270, 315\}$ degrees (0° = Habitat North). This generates between 3–7 FPVs from the agent at each location l_i. More formally, the set of yaw rotations that we use: $YR \subseteq PYR$ such that $3 \leq |YR| \leq 7$. Therefore, the set of all evaluated FPVs in a given habitat: $S_{FPV} = \{l_i \times YR_i\}_{i=1}^{N}$. This is repeated for H habitats, in this study $H = 25$:

$$\begin{aligned} S_{AL} &= \{\{l_{i,h} \times YR_{i,h}\}_{i=1}^{N_h}\}_{h=1}^{H} \\ S_{AFPV} &= \sum_{h=1}^{H}(\{l_{i,h} \times YR_{i,h}\}_{i=1}^{N_h}) \end{aligned} \tag{2}$$

where S_{AL} is a set of sets that contain all FPVs grouped for each explored habitat; S_{AFPV} is the set of all FPVs that is explored across all habitats (without segregating them by habitat); and N_h is the number of locations in the h-th habitat.

We recognise that in a physical setting, robot exploration of the habitat would be via a random walk whereby each FPV would be evaluated in sequence. Adopting the teleportation option in the AI2-THOR simulator for expediency leads to fewer objects observed compared to a sequential random walk. To compensate, we choose a high number of locations (and robot orientations) in each habitat, while keeping the minimum separation between the locations at 1.25 m. We set the visibility distance to 3 m to allow the agent to observe further objects, and we maintain the camera at 30 degrees tilted below the horizon. This leads to an appropriate density of locations with sufficient numbers of observable objects for a valid demonstration of utility.

Algorithm 1 Habitat Exploration

Input: None

Output: Gridmap, Room Type, Observable objects and Inference Time at each visited location

```
Select random locations to visit
for location in selection do
    Teleport to the location
    Store location and observable objects
for location in selection do
    Infer room type with LLM based on observed objects
    Infer room type with SVC based on observed objects
    Store both inference results and time taken to infer
return Set of Classified locations with observed objects, inference times and generated gridmap
```

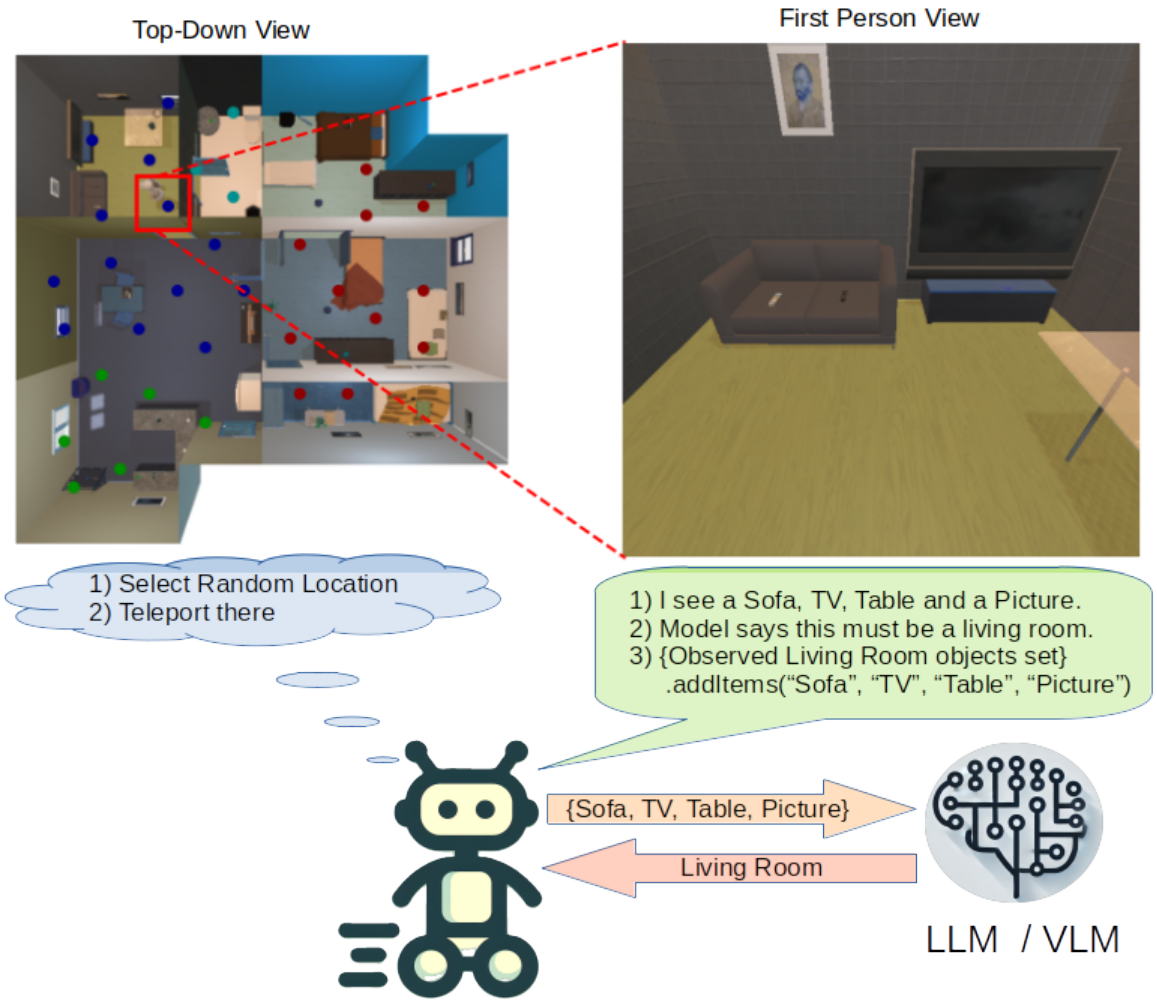


Figure 3. Habitat exploration phase. Agent teleports to random locations and infers the type of room based on observed objects. The room type is inferred by prompting an LLM / VLM.

3.3.1. Objects used for classification

Of the objects that may be observed in each FPV, we class the following as common objects which means they are not semantically useful to uniquely identify a room: *{'Wall', 'Doorway', 'Window', 'Floor', 'Doorframe'}*. Occasionally the set of observed objects at a location would consist of common objects only, as such we call such sets Hard To Classify (HTC) and do not use them for room classification with LLM. We do, however, attempt room classification when the object set includes at least one non-common object.

The difficulty of assigning a room label to an object set differs between sets and is generally related to the size of the set, the proportion of common objects that it contains and the semantic relationships between objects and room types. It is therefore important to recognize that some object sets are more confusing than others. Whilst we have not quantified the confusion score of object sets, we do need to recognize confusing sets as such and introduce a term for them: High Confusion Level (HCL) sets. Classifying HCL sets may lead to lower accuracy, although in general enough semantic information exists within them to allow for a classification that is better than random guess. The important difference between HTC and HCL sets is that for room classification we do not use HTC sets with LLM while HCL sets are always used. For VLM room classification we use both HTC and HCL sets as common objects

may contain visual features that are semantically informative as described below.

3.3.2. Room classification experiment using LLM

At the end of the exploration phase, we use the LLMs to classify each FPV into one of the room types to answer the question: "How is room classification accuracy affected by the choice of an LLM?" The independent variable being choice of LLM (Table 1) and the dependent variable being accuracy of room classification across all FPVs in sets S_{AL} and S_{AFPV} as defined in Equation (2). The controlled variables are the S_{AL} and S_{AFPV} sets, and the hardware used.

We use the collected list of objects at any given FPV and query an LLM for the room type. To each FPV we assign the tag of *bathroom*, *kitchen*, *bedroom* or *living room* based on the objects observed, e.g., from point A in Figure 4a the following objects might be observed: *vase, shelving unit, sink basin, garbage can*. When LLM is prompted with a question of what kind of room would contain these objects, it is likely to respond that the room is bathroom. Similarly, point B would be recognized as a kitchen based on observed: *fridge, worktop, lettuce and apple*. See Figure 2a for the LLM prompt used.



(a) **(b)**

Figure 4. **(a)** Top view of an example habitat with two reference points A and B. Point A should be classified as Bathroom based on observed vase, shelving unit, sink basin and garbage can. Point B should be classified as kitchen based on observed fridge, worktop, lettuce and apple; **(b)** FPV of a living room within AI2-Thor habitat simulator which was misclassified as bedroom by LLM.

3.3.3. Room classification experiment using VLM

Our approach to room classification using LLMs is inherently limited as we only use lists of objects that have been identified at given locations and ignore other salient clues such as the placement of objects. Two semantically different rooms may contain the same objects leading to mis-classification by an LLM. For example, the following set of objects: "{*Wall*, *DogBed*, *Pillow*, *Sofa*, *Floor*}" would typically lead to an assignment of *bedroom* by an LLM. However, in this particular example (see Figure 4b) the ground truth was *living room* with the *DogBed* and *Pillow* objects biasing the LLM towards the *bedroom* label.

Further, we can not use LLMs for HTC object sets as described in Section 3.3.1. The visual appearance of common objects may still present useful clues, e.g. a floor or wall may be tiled, which would suggest a bathroom or kitchen as opposed to bedroom or living Room. This motivated us to repeat the LLM experiment as defined in Section 3.3.2, using VLM, which for our purposes are models that accept digital images along with text in the prompt.

We perform the room classification task with VLMs using the same set of locations and FPV orientations as used for the LLMs but also include the HTC locations as explained in Section 3.3. The input presented to the VLMs from each location was the actual FPV image from the agent instead of a list of detected objects as for the LLMs. For a fair comparison between LLM and VLM, we introduced three classification metrics for VLM performance: **with HTC**, **without HTC** and **HTC only**. For reasons explained in Section 3.2.2, we also introduced additional metrics to accommodate:

- **4 label output**, **6 label output** and unspecified label count referenced as **0 label count**
- **With CoT** and **without CoT**
- With image placeholder located in the **middle** and **end** of the prompt
- Quantization: **16bit** and **4bit**

The relevant combinations of these metrics lead to modes of classification as defined in Table 5. We also compare the VLM performance against Llama3 as it was found to be the best perfoming LLM.

3.3.4. Room classification experiment using VLM and LLM hybrid approach

Results from the VLM and LLM experiments suggest an approach to improve overall performance of room classification by combining the 2 models. Specifically, there are merits to using an LLM for non-HTC views and a VLM for HTC views. We chose Llama3 for the LLM as it exhibited the best results in among all evaluated LLMs. To compare the proposed hybrid methods with the Llama3 results, we had to re-evaluate Llama3 result by including HTC object sets. This could be done in two ways, either treating HTC sets automatically as a classification failure (HTC discounted) or randomly guessing the room type for HTC sets (HTC guessed). For hybridization of the VLM and Llama3 results, we again have to take into account the peculiarities of the VLM models, as explained in Section 3.2.2. This leads to Llama3 evaluations and hybridization modes of VLM and Llama3 results as defined in Table 6.

3.3.5. Room classifier framework and scoring

To formalize the room classification scoring for experiments and methods described in Sections 3.3.2, 3.3.3 and 3.3.4, we define three classification functions: F_{LLM}, F_{VLM} and F_{HYB} which classify an FPV using LLM, VLM or the hybrid approach respectively:

$$F_{LLM}(FPV) = \begin{cases} 0, & \text{if } FPV \in S_{HTC}. \\ 1, & \text{if } LLM(FPV) = GT_{FPV}. \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

$$F_{VLM}(FPV) = \begin{cases} 1, & \text{if } VLM(FPV) = GT_{FPV}. \\ 0, & \text{otherwise.} \end{cases} \tag{4}$$

$$F_{HYB}(FPV) = \begin{cases} F_{VLM}(FPV), & \text{if } FPV \in S_{HTC}. \\ F_{LLM}(FPV), & \text{otherwise.} \end{cases} \tag{5}$$

where $LLM(FPV)$ runs LLM inference on the FPV, $VLM(FPV)$ runs VLM inference on the FPV, S_{HTC} is a set of all HTC scenes and GT_{FPV} is ground truth for the particular FPV.

All FPVs in a given habitat are classified using the classification function F_{CLS}, which is defined as one of F_{LLM}, F_{VLM} or F_{HYB} depending on the use case.

$$F_{CLS}(FPV) = \begin{cases} F_{LLM}(FPV), & \text{if LLM is being tested.} \\ F_{VLM}(FPV), & \text{if VLM is being tested.} \\ F_{HYB}(FPV), & \text{if Hybrid approach is being tested.} \end{cases} \tag{6}$$

This essentially compares the LLM or VLM output to ground truth.

We can now define a scoring function for a habitat. The number of correct classifications is divided by the total amount of FPVs analyzed to obtain the mean accuracy score per habitat, or more formally: let S_{FPV} be the set of FPVs in a habitat as shown in Section 3.3, then

$$HS(S_{FPV}) = \frac{\sum_{j=1}^{|S_{FPV}|} F_{CLS}(S_{FPV}[j])}{|S_{FPV}|} \tag{7}$$

where $HS(S_{FPV})$ is a habitat score and $S_{FPV} \in S_{AL}$ from Equation (2).

We can calculate scores for the entire S_{AL} set: $HS_{AL} = \{HS(S_{FPV_i})\}_{i=1}^{|S_{AL}|}$, where $S_{FPV_i} \in S_{AL}$ for such i that $1 \leq i \leq |S_{AL}|$.

Let $\overline{HS}_{AL}$ be a list derived from HS_{AL} set and sorted in ascending order.

We can now define two classifier scores to evaluate our classification method:

$$CS_{mm} = \begin{cases} \overline{HS}_{AL}[\frac{n+1}{2}], & \text{if } n \text{ is odd.} \\ \frac{\overline{HS}_{AL}[\frac{n}{2}] + \overline{HS}_{AL}[\frac{n}{2}+1]}{2}, & \text{if } n \text{ is even.} \end{cases} \tag{8}$$

where $n = |S_{AL}|$

$$CS_{om} = HS(S_{AFPV}) \tag{9}$$

CS_{mm} is essentially the median score of all habitat scores, which in turn are mean scores of all FPVs in each habitat. CS_{om} is the ordinary mean score calculated over the set S_{AFPV} from Equation (2) without segregating the FPVs by habitats. CS_{om} provides a good overview of mean accuracy across all habitats, while CS_{mm} minimizes the influence of outliers.

As a final note, for LLM experiments, we adjust sets S_{AL} and S_{AFPV} to not contain HTC FPVs as discussed in Section 3.3.1.

3.4. Goal selection experiments

Goal selection is outlined in Algorithm 2 and illustrated in Figure 5. We let the human operator define the target object to be retrieved which can be given in free text and does not have to belong to a defined dictionary or to have been previously observed, for example, the target object might be a bottle of beer. The LLM is prompted with a question of where would it be most likely to find a bottle of beer and given all detected room types to select from (prompt 2c). The LLM should respond that kitchen is the best candidate. A list is then compiled of all the objects that have been observed in the kitchen during the exploration phase, and a new prompt is prepared for the LLM asking which of the objects in the list should be explored first if we are looking for a bottle of beer (prompt 2b). In the given case, the LLM is likely to suggest fridge. The position of fridge is known from the exploration phase when it was observed, so a path is planned to navigate to fridge from the current location of the agent. For path planning, we only used pre-generated gridmap and an A* algorithm implementation- both available in Thortils, which we applied to locations of objects directly observed in AI2-THOR simulation.

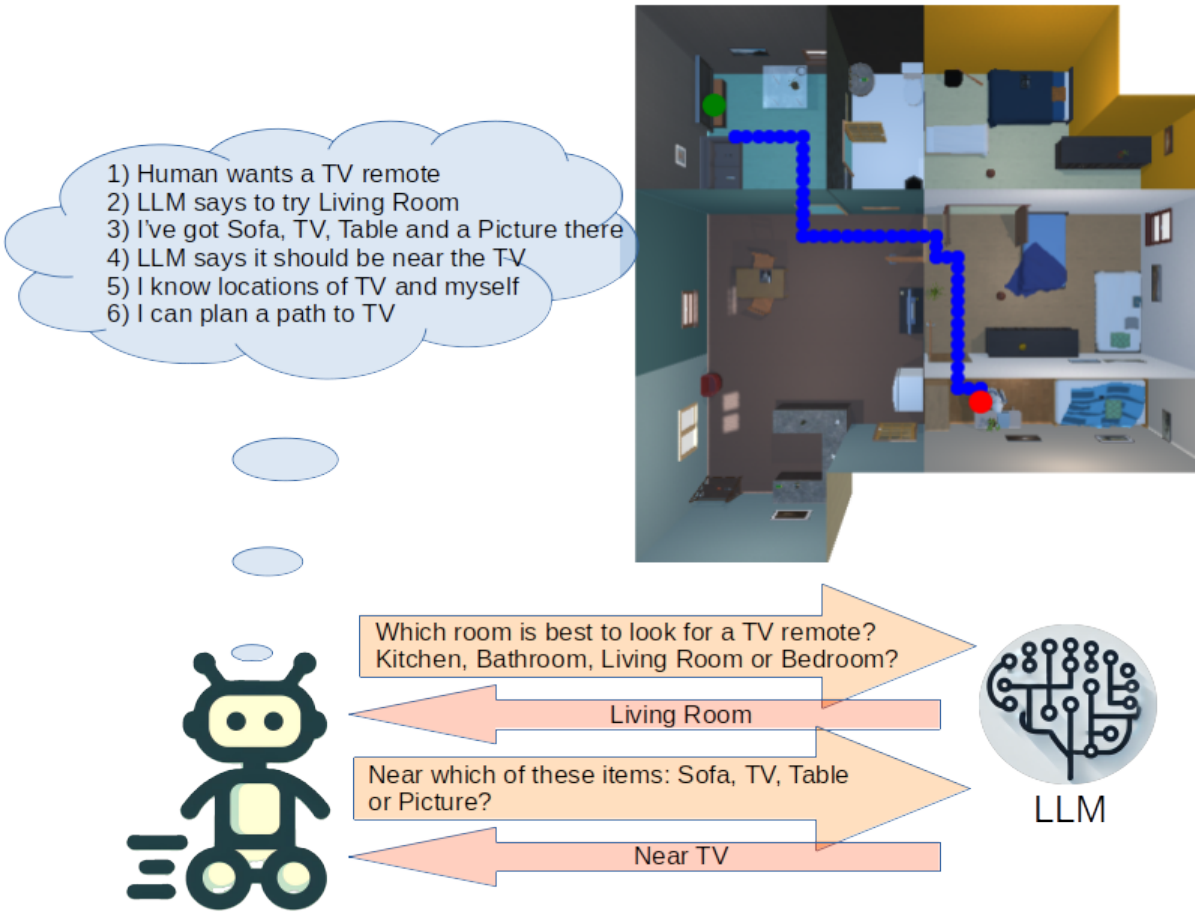


Figure 5. Semantic navigation phase. The required room is inferred from user's request using an LLM. Then a list of the observed objects at the required room is compiled. The list is passed as part of a further prompt to the LLM in order to identify the object that was requested by user or the most likely goal that will be closest to the requested object. A path is planned to the identified goal.

Algorithm 2 Goal Selection

Input: Target to find in free-text

Output: Executable path to the target

```
room = Query_LLM_for_which_room_to_find_target()
object_set = Look_up_all_objects_in(room)
object = Query_LLM_for_which_object_is_target_likely to_be_near(object_set)
path = Plan_path_to_location_of(object)
return path
```

3.4.1. Qualitative observation experiment

For this experiment we chose the target object from the following list: *Bottle of Beer, Car Keys, Credit Card, Hair Pin, Pillow, Pyjamas, Toothbrush, TV Remote* and only test the Llama3 LLM to select goals and generate paths. The performance is then qualitatively analyzed and discussed. We conducted this experiment in a single, randomly selected habitat from ProcTHOR-10k dataset, specifically habitat 55 in the "train" collection.

3.4.2. Quantitative analysis experiment

For this experiment, we selected 100 common household items and grouped them into 4 lists corresponding to room categories (*bathroom, kitchen, bedroom or living room*). This lead to 25 items per room type. Please see Appendix 2 for details. We then performed room and goal selection for each item repeating it 100 times. This lead to 100 goal selection test samples (selected room and object) per item. We then categorized results from each of the test samples against 3 types of error and assigned a score based on which, if any, of these errors occurred:

1. **TYPE1: No room selected.** This would happen if none of the 4 room classes were found in the output of the LLM while prompting for room selection (Figure 2c).
2. **TYPE2: No object selected.** This would happen if no object from the given list was found in the output of the LLM while prompting for object selection (Figure 2b)
3. **TYPE3: The selected room did not match the room label of the selected object.** This would happen if LLM did not agree to our assignment of items per room type.

If a TYPE1 error is present in a test sample, then it scores 0 points. If TYPE2 error present then 1 point awarded, if TYPE3 then 2 points, else no errors is scored as 3 points. Note that the error types are not exclusive; the presence of TYPE1 error also implies TYPE2 and TYPE3 since no object can be selected and no room can be compared to the expected one.

Thus the set of scores per 1 household item is $IS = \{s_n\}_{n=1}^{100}$ where s_n represents the score of the *n-th* test sample. Since we assigned 25 items to each room, the total set of scores per room is $RS = \sum_{i=1}^{25}(\{s_{n,i}\}_{n=1}^{100})$ where i is the index of the item in a given room and n is the index of the test sample of the *i-th* item.

We then calculate the mean score S_m (Equation (10)) and standard deviation σ (Equation (11)) across the RS set for each room.

$$S_m = \frac{\sum RS}{|RS|} \tag{10}$$

$$\sigma = \sqrt{\frac{\sum_{c=1}^{|RS|}(\overline{RS}[c] - S_m)^2}{|RS| - 1}} \tag{11}$$

where $|RS| = 2500$ for our purposes and $\overline{RS}$ is a list of room scores derived from RS set.

Finally, we analyze the room selection vs our assigned room and presence of the aforementioned error types in test samples for all items. We perform this analysis for all LLMs that we used. This experiment was conducted in the same single habitat as the Qualitative Observation experiment in Section 3.4.1, specifically habitat 55 in the "train" collection of ProcTHOR-10k habitats' dataset.

4. Results

We processed 25 habitats from the ProcTHOR-10k dataset, as described in Section 3.3, with each habitat containing at least one instance of every room type: bathroom, kitchen, bedroom and living room. The number of rooms per habitat ranged from 4 to 10 ($mean = 6.24$, $median = 5$). Each habitat was sampled between 64 and 243 times (FPV capture) depending on its size, resulting in a total of 3639 FPVs classified by the LLMs and the control SVC (S_{AL} in equation 2). The VLM model data set ($S_{AL} = 4245 FPVs$) included the 606 HTC scenes which were removed from the LLM set. The number of objects at each FPV ranged between 1 and 20 ($mean = 5.22$, $median = 4$).

4.1. Room classification results

The room type for each FPV was inferred and compared with the ground truth, as described in Section 3.3.5, leading to habitat accuracy scores for each classification method as per Equations (7)–(9). These habitat accuracy scores are shown in Figure 6 as box plots with the inference times for each model summarised in Table 4.

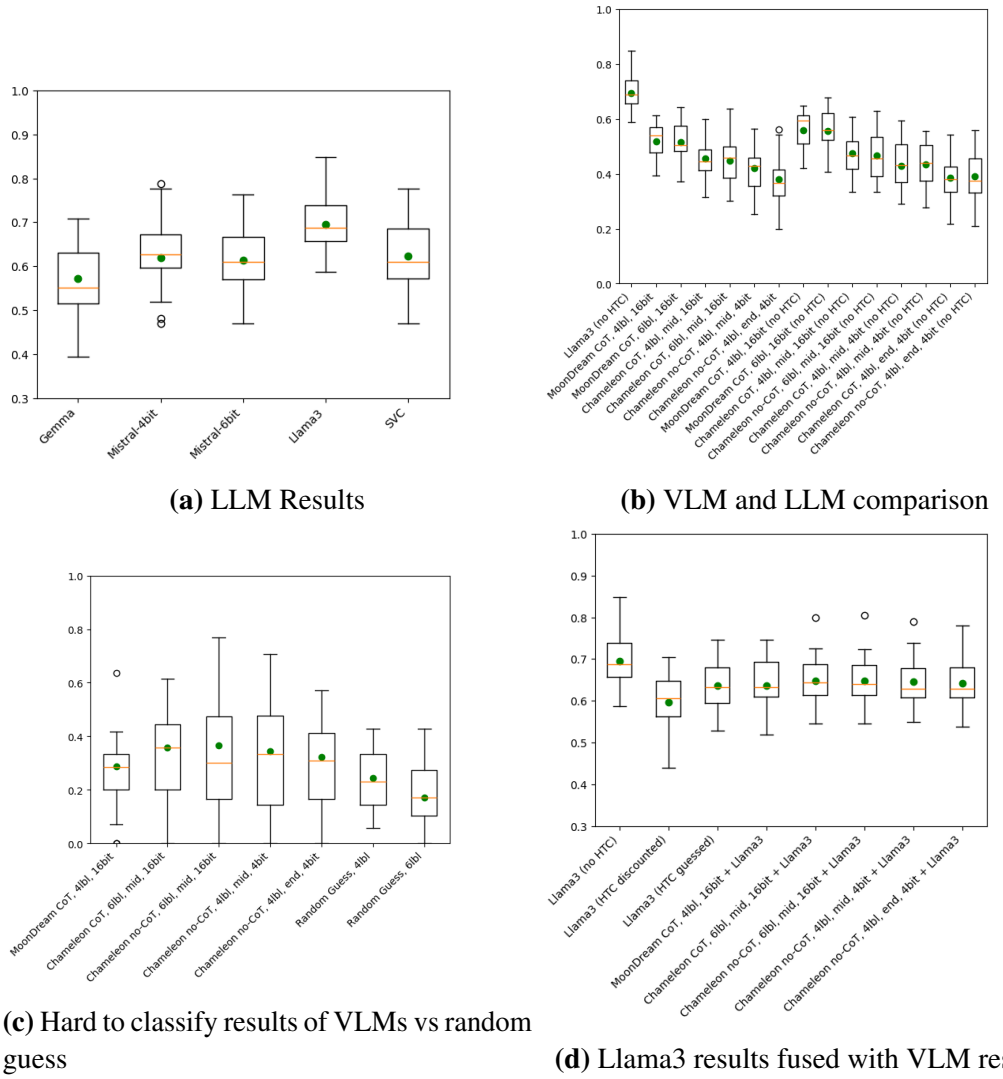
Table 4. Inference Time.

Model	**Inference Time (s)**		
	Min	Max	Mean
Gemma	1.89	29.71	5.45
Mistral 4 bit	0.92	25.48	4.88
Mistral 6 bit	2.50	74.08	13.82
Llama 3	4.59	55.34	15.48
SVC	0.001	0.004	0.001

4.1.1. Results of room classification using LLM

As is clear in Figure 6a, all LLMs performed significantly above the random base line of 0.25 (4 room classes). Gemma LLM is outperformed by all other LLMs and the SVC. The performance of both Mistral

LLM versions is similar to that of the SVC, whilst Llama3 LLM outperforms all other LLMs and SVC. We believe the common variance of approximately 0.3 for all models can be largely accounted for by the presence of HCL object sets as described in Section 3.3.1. In addition, we noted that some FPVs contained views of objects through open doorways into adjoining rooms resulting in misclassifications. Finally, the LLM models do not always follow the instruction to provide an easily interpretable answer as explained in Section 3.2. This may lead to mis-interpretation of the answer and also contribute to the observed variance.



(a) LLM Results

(b) VLM and LLM comparison

(c) Hard to classify results of VLMs vs random guess

(d) Llama3 results fused with VLM results

Figure 6. Room classification results using different methods. The box plots show variances of the accuracy scores per method: the box represents variance of 50% of the accuracy scores while another 25% of the scores is above the box and 25%- below- both marked with the whiskers, while the empty circles show outliers. The orange line within the box shows the median score as defined in Equation (8), the green circle shows mean score as defined in Equation (9). For (b), (c) and (d): Each box plot is labelled with the name of the model, whether CoT was used, the number of labels, the placement of image placeholder and quantization bits.

Interestingly, Mistral LLM with weights quantized to 4-bits marginally outperforms its 6-bit quantized version. This could be explained by the 4-bit quantized model being "lucky" in HCL object set selection, however, the upper whisker is more established which raises the question as to

whether the 6-bit quantization is optimal for the proposed purpose.

We also see that the mean inference time for 6-bit quantized LLM versions correlates with mean accuracy, but the 4-bit quantized Mistral LLM breaks this trend (Table 4). This further supports the question of whether 6-bit quantized versions of LLM are optimal for the proposed purpose as Mistral 4-bit is both more accurate and faster than Mistral 6-bit. The correlation for the 6-bit quantized LLMs however is strong and suggests that the architecture of the LLM plays significant role for the proposed purpose.

As expected, the SVC prediction time is 3 to 4 orders of magnitude better than any of the LLMs with comparable accuracy, with only Llama3 consistently outperforming the SVC in accuracy. This supports the use of SVC in room classification as suggested in [24] but only if speed versus accuracy compromise is acceptable. It is also noteworthy that the slow response times of LLMs are due to the justification text that they generate. This is addressed in the field by optimizing prompting or parsing the text before all of it has been generated.

Finally, the difference between minimum and maximum inference times is quite large for all LLMs. This stems from the inference speed (tokens per second) slowing down over time during habitat exploration. We did not investigate this phenomenon in greater detail, but stability issues of current software releases should certainly be a factor when considering use of LLM for the proposed purpose. We expect, however, these issues to be addressed as the technology matures.

4.1.2. Results of room classification using VLM

Table 5 shows the accuracy of room classification performed by the VLMs Chameleon and Moondream2 in their various test modes, compared with LLama3 which was the best performing LLM. All Chameleon and Moondream2 results for classifying non-HTC FPVs are markedly lower than Llama3, though still above random chance. The VLM results for non-HTC FPVs are consistently worse than Llama3 results irrespective of the 4-room or 6-room label cases, although Moondream2 clearly outperforms Chameleon. The best performing classification modes are shown as box plots in Figure 6b. Interestingly, the image placeholder placement in the middle of the prompt for Chameleon is clearly advantageous. Also, the CoT request to explain itself is only advantageous in 16-bit quantization, but not in 4-bit. Nevertheless, this suggests that the overall reasoning capability of our tested VLMs is inferior to that of Llama3. Nevertheless, a noteworthy observation is that several classification modes led to HTC results above random guess baseline, which suggests the use of VLM for HTC cases only and Llama3 for non-HTC cases. This is further supported in Figure 6c, where best HTC results are shown as box plots along with random guess baseline.

4.1.3. Results of room classification using VLM and LLM hybrid approach

The mean and median scores of the explored hybrid and LLM methods are given in Table 6 and shown in Figure 6d. The Llama3 (HTC guessed) method approximates a Llama3 score in a real-world scenario since HTC scenes are a common occurrence and cannot be reliably analyzed using LLM. Un-quantized Chameleon model, when used with 6 labels, clearly improves this result by 1% and 2% of median and mean scores respectively. Even the contribution of 4-bit quantized Chameleon model improves Llama3 (HTC guessed) mean score, although in a different classification mode (using 4 labels instead of 6 and not

using CoT). Contribution of Moondream2 is less pronounced. Importantly, these contributions rely on the number of HTC FPVs present. The incidence of HTC FPVs in our dataset reached circa 14%. A higher incidence of HTC FPVs would lead to a higher impact of VLM contribution, thus raising the scores in Table 6 for the hybrid methods.

Table 5. VLM performance. "CoT" indicates whether the VLM was asked to provide reasoning for its answer. "lbl cnt" is number of labels permitted in the prompt. "img loc" indicates the location of image placeholder. "Q" is number of bits used in quantization. "All" shows results of all FPVs (including HTC). "HTC" shows results of HTC FPVs only. "nHTC" shows results of non-HTC FPVs only. "mm" indicates median score, which was calculated using Equation (8). "om" indicates mean score, which was calculated using Equation (9). "Prompt Used" references prompts from Table 3 and Figure 2a. Results are grouped by VLM type, Image Location and Quantization. The highest scores in each group are highlighted bold. Prompt P15 was evaluated within the group of 16-bit quantized Chameleon model with image placeholder in the middle due to the structure of the prompt. Llama3 LLM results are included at the end of the table for comparison.

Model	**CoT**	**lbl**	**img**	**Q**	**All**$_{mm}$	**All**$_{om}$	**HTC**$_{mm}$	**HTC**$_{om}$	**nHTC**$_{mm}$	**nHTC**$_{om}$	**Prompt Used**
MoonDream2	yes	0	NA	16	0.50	0.50	0.18	0.23	0.56	0.54	P1
MoonDream2	yes	4	NA	16	**0.54**	**0.52**	**0.29**	**0.29**	**0.59**	**0.56**	P2
MoonDream2	yes	6	NA	16	0.51	**0.52**	0.21	0.27	0.56	**0.56**	P3
MoonDream2	no	0	NA	16	0.40	0.39	0.09	0.12	0.44	0.44	P4
MoonDream2	no	4	NA	16	0.50	0.50	0.18	0.24	0.55	0.54	P5
MoonDream2	no	6	NA	16	0.49	0.48	0.24	0.22	0.53	0.52	P6
Chameleon	yes	0	Mid	16	0.36	0.37	0.21	0.23	0.39	0.39	P7
Chameleon	yes	4	Mid	16	0.45	**0.46**	0.33	0.34	**0.47**	**0.47**	P8
Chameleon	yes	6	Mid	16	**0.46**	0.45	**0.36**	0.36	0.44	0.46	P9
Chameleon	no	0	End	16	0.34	0.36	0.20	0.23	0.35	0.38	P15
Chameleon	no	4	Mid	16	0.34	0.36	0.18	0.24	0.37	0.38	P10
Chameleon	no	6	Mid	16	0.45	0.45	0.30	**0.37**	0.46	**0.47**	P11
Chameleon	yes	0	Mid	4	0.31	0.32	0.20	0.23	0.33	0.34	P7
Chameleon	yes	4	Mid	4	0.40	0.41	0.27	0.31	0.43	**0.43**	P8
Chameleon	yes	6	Mid	4	0.39	0.38	0.20	0.23	0.39	0.40	P9
Chameleon	no	4	Mid	4	**0.43**	**0.42**	**0.33**	**0.34**	**0.44**	**0.43**	P10
Chameleon	no	6	Mid	4	0.39	0.39	0.29	0.30	0.41	0.41	P11
Chameleon	yes	0	End	4	0.30	0.31	0.21	0.20	0.33	0.33	P12
Chameleon	yes	4	End	4	**0.37**	0.37	0.25	0.28	**0.38**	0.38	P13
Chameleon	yes	6	End	4	0.29	0.30	0.17	0.22	0.31	0.31	P14
Chameleon	no	0	End	4	0.33	0.33	0.20	0.23	0.34	0.35	P15
Chameleon	no	4	End	4	**0.37**	**0.38**	**0.31**	**0.32**	**0.38**	**0.39**	P16
Chameleon	no	6	End	4	0.23	0.25	0.12	0.17	0.25	0.26	P17
Llama3	yes	4	NA	6	NA	NA	NA	NA	**0.69**	**0.70**	Fig. 2a

Table 6. Room classification results with hybrid methods. Median score is calculated using Equation (8). Mean score is calculated using Equation (9). VRAM is the combined memory footprint for operating the given models.

Method	Median Score	Mean Score	VRAM
Llama3 (no HTC)	0.69	0.70	6.74GB
Llama3 (HTC discounted)	0.61	0.60	6.74GB
Llama3 (HTC guessed)	**0.63**	**0.63**	6.74GB
MoonDream CoT, 4lbl, 16bit + Llama3	0.63	0.64	10.92GB
Chameleon CoT, 6lbl, mid, 16bit + Llama3	**0.64**	**0.65**	22.47GB
Chameleon no-CoT, 6lbl, mid, 16bit + Llama3	**0.64**	**0.65**	22.47GB
Chameleon no-CoT, 4lbl, mid, 4bit + Llama3	0.63	**0.65**	10.67GB
Chameleon no-CoT, 4lbl, end, 4bit + Llama3	0.63	0.64	10.67GB

4.2. Results of goal selection

We performed goal selection experiments as described in Section 3.4 and present the results here. We first provide the qualitative analysis of the results of experiment described in Section 3.4.1 and then the quantitative analysis of the results of experiment described in Section 3.4.2.

4.2.1. Qualitative analysis

We present eight examples of finding everyday items in a habitat in Figure 7. For each target, the LLM first selected the most likely room to find it and then the most likely object it could be next to from a list of observed objects in that room. This is summarized in Table 7. In all cases a reasonable justification was given by the LLM and we used Llama3 for all examples. Then a path was generated using A* algorithm. Curiously, the credit card target in Figure 7c led us to the kitchen counter top, but among other suggested locations was the fridge. Although the fridge may not be everybody's choice for keeping a credit card, a reasonable justification was given in favor of that location. This demonstrates that SN decisions can be ambiguous, but it is encouraging that LLM is able to suggest reasonable, albeit not always obvious locations. The hair pin target in Figure 7d was supposed to lead to a dresser in the bedroom, but instead led to a dresser in the bathroom. This is because one or more random locations in the bathroom were misclassified as bedroom in the exploration phase and the wrong dresser was selected. Similarly, pillow in Figure 7e was supposed to be found in the bedroom, but instead was found in the living room. This too was due to the mis-classification of room, but it did find the actual target that was requested. This shows that room mis-classification for a limited number of locations may not always lead to wrong navigation decisions and in this case actually led to a more optimal destination as the pillow in the living room was nearer than the one in bedroom. Finally, Pyjamas in Figure 7f were sought on a bed in the bedroom and the destination location is close enough to the bed, alas behind a wall. This was due to the A* algorithm selecting a suboptimal destination point and can be improved with smarter path generator algorithms. The remaining 4 examples do not elicit comments.



(a) Bottle of Beer.



(b) Car Keys.



(c) Credit Card.



(d) Hair Pin.



(e) Pillow.



(f) Pyjamas.



(g) Toothbrush.



(h) TV Remote.

Figure 7. Goal was chosen by Llama3 LLM and path was generated using A* algorithm to find each of the items. Path in blue colour is overlaid on the top-view of the habitat. Red dot is the starting location and green dot- the goal location of the path.

Table 7. Navigation targets.

Target	Selected Room	Selected Object
Bottle of Beer	Kitchen	Fridge
Car Keys	Bedroom	Desk
Credit Card	Kitchen	Counter Top
Hair Pin	Bedroom	Dresser
Pillow	Bedroom	Pillow
Pyjamas	Bedroom	Bed
Toothbrush	Bathroom	Sink
TV Remote	Living Room	Television

4.2.2. Quantitative analysis

The decisions made by LLM do not lend themselves well to quantitative analysis, e.g., the goal for car keys was selected to be the desk in the bedroom while for credit card it was the counter top in the kitchen. Both are reasonable selections from a human perspective, however, the goals for these items could have easily been swapped and would be just as reasonable. This makes it difficult to evaluate goal selection choices as simply correct or incorrect. Similarly, it is open to interpretation whether example in Figure 7e was a successful goal selection or not as explained in Section 4.2.1. Assigning a mean success rate to the set of examples in Section 4.2.1 can lead to anything between 50% and 100% depending on our own subjective criteria and understanding of the world and to what extent it agrees with the LLM.

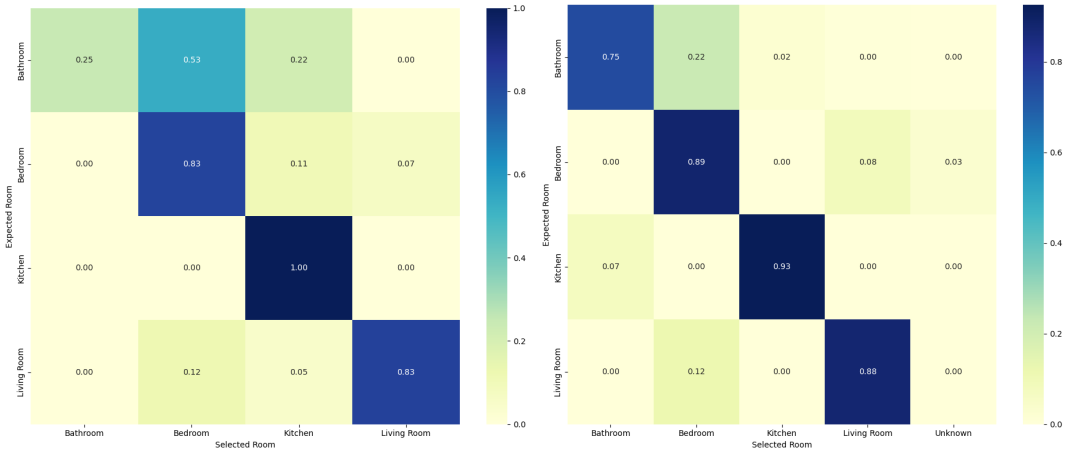
We can, however, analyze the consistency of the choices the LLM makes. Using the scoring mechanism that we introduced in Section 3.4.2, we captured mean and standard deviation of scores per room type per LLM in Table 8. We also present a summary of errors (as defined in Section 3.4.2) made by LLMs in Table 9.

Table 8. Goal selection mean scores and standard deviation per room per LLM. Mean is calculated using Equation (10), Std is calculated using Equation (11). Room labels are the assigned group labels to the household items as explained in Section 3.4.2.

Model	Kitchen		Bathroom		Living Room		Bedroom	
	Mean	Std	Mean	Std	Mean	Std	Mean	Std
Llama 3	2.92	0.27	2.73	0.47	2.87	0.34	2.79	0.61
Mistral 6 bit	2.94	0.24	2.84	0.37	2.89	0.34	2.63	0.74
Mistral 4 bit	2.94	0.23	2.82	0.40	2.88	0.37	2.71	0.61
Gemma	3.00	0.00	2.25	0.44	2.83	0.37	2.81	0.40

The mean scores are approaching or at maximum in Table 8 and deviation is mostly modest, which indicates encouraging performance of goal selection accross all tested LLMs. For all models, except Gemma, the goal selection is least consistent (*i.e.* deviation is highest) in Bedroom and most consistent in Kitchen. Gemma, on the other hand, achieves a perfect score in Kitchen, but its mean score underperforms

in Bathroom compared to other models. Mistral 6-bit quantized model, on the other hand, is the best performer in Bathroom and Living Room. Interestingly, the 4-bit quantized Mistral model achieves very similar results to its 6-bit quantized version in 3 out of 4 rooms, while clearly outperforming it in Bedroom. Similar superiority was observed in room classification performance as explained in Section 4.1.1. This further supports the use of 4-bit quantization. Furthermore, both Mistral LLMs clearly outperform Llama3 in Kitchen and Bathroom while Llama3 and particularly Gemma excel in Bedroom. This is further supported by the summary of the observed errors in Table 9. Finally, Type3 error relates to how well our assigned (*i.e.* expected) room labels match the LLM predicted ones. The presence of this error differentiates the results of Llama3 and Gemma in Bedroom, where even though Gemma scores better overall, its error incidence is heavily skewed towards Type3. This is further illustrated with a confusion matrix between rooms for Gemma in Figure 8a. Llama3 and especially Gemma suffers from confusion between Bathroom and Bedroom while both Mistral quantizations have more stable performance overall as is shown in Figure 8.



(a) Gemma Room Confusion Matrix. **(b)** Llama3 Room Confusion Matrix.

(c) Mistral 4-bit Room Confusion Matrix. **(d)** Mistral 6-bit Room Confusion Matrix.

Figure 8. Room confusion matrices per LLM tested. Each matrix shows the incidence of the expected room matching the selected room and bias towards others across 10000 test runs per LLM (2500 per room per LLM).

Table 9. Total error counts across all items per room per LLM. T1, T2 and T3 denote error types as described in Section 3.4.2. Room labels are the assigned group labels to the household items. The total number of samples are 2500 per room per LLM.

Model	Kitchen			Bathroom			Living Room			Bedroom		
	T1	T2	T3	T1	T2	T3	T1	T2	T3	T1	T2	T3
Llama 3	0	14	186	5	44	629	2	7	309	74	178	286
Mistral 6 bit	0	50	107	0	19	386	6	77	201	126	180	612
Mistral 4 bit	0	30	112	0	51	410	13	51	225	70	125	523
Gemma	0	0	0	0	4	1877	0	0	417	0	50	413

5. Conclusion & future work

We have evaluated the use and performance of local quantized LLMs for room classification and goal selection in a domestic SN setting.

We carefully chose the parameter count and quantization levels to allow a combined memory footprint of an LLM and a VLM model to not exceed 12 GB of VRAM in most cases (see Table 6). This allows for the use of these models on mobile platforms (e.g. NVIDIA Jetson) or a commercial grade laptop. More specifically, we used NVIDIA Jetson AGX Xavier, which has 32GB of combined memory, but more limited platforms are possible, e.g. Nvidia Jetson ORIN NX, which has 16GB of combined memory. Alternatively, any x86 architecture computer with a minimum of 12GB of VRAM should also be suitable (although was not tested).

We compared various LLMs with two different quantization levels (4-bit and 6-bit) and have demonstrated that room classification accuracy of LLMs is comparable to a much faster SVC solution, with the latest architecture improvements (Llama3) outperforming SVC in accuracy. We also found that 4-bit quantization may perform comparably to 6-bit if not better. This is a direction worth exploring in future work, as it is somewhat counter-intuitive and unexpected. This also motivates the creation of purpose-trained LLMs for SN- that should reduce the size and increase inference speed while keeping the flexibility of the solution.

We have also explored the utility of VLMs in SN, specifically, Chameleon and Moondream2. Individually their results at room classification were worse than Llama3, however, we have shown that VLMs can be used as a complement to LLMs to improve HTC scene classification. Implementation of such a hybrid solution in a real-world agent will be an important direction for future work. Furthermore, deeper semantic analysis can be applied to the FPVs, for example, an LLM can be prompted with a history of previously seen objects while traveling alongside the newly observed set. This would only be possible with continuous habitat exploration, unlike our teleportation approach, but it could lead to significant improvements in accuracy and a more elegant solution to the HTC problem.

The varying performance of measured LLMs in goal selection task, highlights the importance of choosing the right LLM for the right room. It also suggests that use of low parameter count, quantized LLMs in SN will likely be best within a framework that switches the LLM according to context. We have

shown a method for making such LLM choices for future use in SN.

Finally, we have demonstrated that a small LLM can also be used to interpret Human language cues to direct the generation of spatial pathways through a multi-room domestic environment that has been randomly explored and semantically labeled using the hybrid LLM-VLM approach.

Acknowledgments

The work of A. Elksnis was supported by the EPSRC Centre for Doctoral Training in Future Autonomous and Robotics Systems - Towards Ubiquity (FARSCOPE-TU), grant number: EP/S021795/1.

Authors' contribution

Arturs Elksnis: Writing–original draft, Conceptualization, Data curation, Formal analysis, Methodology, Visualization, Software; Ziheng Xue: Formal analysis, investigation, Software; Martin Pearson: Resources, Writing-review editing, Supervision; Ning Wang: Conceptualizaiton, Project administration, Supervision, Writing-review editing. All authors have read and agreed to the published version of the manuscript.

Conflicts of interests

The authors declare no conflict of interest.

Appendix

1. Quantization configuration details

The BitsAndBytes configuration that was used for quantization of Chameleon VLM:

Algorithm 3 BitsAndBytes Configuration

```
quantization_config = BitsAndBytesConfig(
        load_in_4bit=True,
        bnb_4bit_quant_type="nf4",
        bnb_4bit_compute_dtype=torch.float16,
        bnb_4bit_use_double_quant=True
)
```

- NF4 quantization (4-bit NormalFloat) for weights.
- *torch.float16* compute dtype for efficient use of the available GPU (exploiting the hardware support for float16 precision as opposed to bfloat16).
- Double quantization (further compressing the quantization constants) to minimize accuracy degradation.

2. Household items used in goal selection task

The household items, that we used for goal selection task, grouped by room class:

Algorithm 4 Household Items by Room Class

```
household_items = {
    "Kitchen": [
        "tea kettle", "food processor", "stand mixer", "rolling pin", "baking sheet",
        "pot holders", "oven mitts", "dish towels", "spice rack", "salt shaker",
        "pepper mill", "cutlery tray", "serving platter", "salad spinner", "ice cream scoop",
        "grater", "thermos", "lunch boxes", "water pitcher", "paper towel holder",
        "trash can", "dish soap", "sponge", "scrub brush", "dish drying mat"
    ],
    "Bathroom": [
        "toothbrush holder", "bath towel", "shower curtain",
        "bath mat", "toilet brush", "plunger", "medicine cabinet",
        "cotton swabs", "nail clippers", "razor", "shaving cream", "hair dryer",
        "curling iron", "makeup mirror", "tissue box", "laundry hamper",
        "facial cleanser", "body wash", "bathrobe", "slippers",
        "magnifying mirror", "electric toothbrush", "floss", "mouthwash", "first aid kit"
    ],
    "Living Room": [
        "sofa", "coffee table", "recliner", "floor lamp",
        "throw pillows", "blanket", "bookshelf", "entertainment center",
        "remote control", "candles", "picture frames", "vase", "clock",
        "ottoman", "side table", "fireplace tools", "chess set", "board games",
        "magazine rack", "curtains", "wall art",
        "DVD player", "gaming console", "throw blanket", "floor cushions"
    ],
    "Bedroom": [
        "nightstand", "dresser", "wardrobe",
        "alarm clock", "bedside lamp", "jewelry box", "full-length mirror", "laundry basket",
        "closet organizer", "shoe rack", "hangers", "duvet cover",
        "blackout curtains", "ceiling fan", "reading chair", "vanity table", "perfume bottles",
        "tissue box cover", "memory foam topper", "electric blanket", "sleep mask",
        "ear plugs", "white noise machine", "closet doors", "underbed storage"
    ]
}
```

References

[1] Borkowski A, Siemiatkowska B, Szklarski J. Towards Semantic Navigation in Mobile Robotics. In *Graph Transformations and Model-Driven Engineering: Essays Dedicated to Manfred Nagl on the Occasion of his 65th Birthday* Engels G, Lewerentz C, Schäfer W, Schürr A, Westfechtel B Eds.,

Heidelberg: Springer Berlin Heidelberg, 2010. pp. 719–748.

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

[3] Crespo J, Castillo JC, Mozos OM, Barber R. Semantic information for robot navigation: A survey. *Appl. Sci.* 2020, 10(2):497.

[4] Anderson P, Wu Q, Teney D, Bruce J, Johnson M, *et al.* Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, Salt Lake City, USA, June 18–23, 2018, pp. 3674–3683.

[5] Ku A, Anderson P, Patel R, Ie E, Baldridge J. Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. *arXiv* 2020, ArXiv:2010.07954.

[6] Moudgil A, Majumdar A, Agrawal H, Lee S, Batra D. Soat: A scene-and object-aware transformer for vision-and-language navigation. In *Proceedings of the 35th Advances in Neural Information Processing Systems*. December 6–14, 2021, pp. 7357–7367.

[7] Tenorth M, Kunze L, Jain D, Beetz M. Knowrob-map-knowledge-linked semantic object maps. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, Nashville, USA, December 06–08, 2010, pp. 430–435.

[8] Gupta R, Kochenderfer MJ, Mcguinness D, Ferguson G. Common sense data acquisition for indoor mobile robots. In *Proceedings of the 19th national conference on Artifical intelligence*, San Jose, USA, July 25–29, 2004, pp. 605–610.

[9] Crespo J, Barber R, Mozos O. Relational model for robotic semantic navigation in indoor environments. *J. Intell. Robot. Syst.* 2017, 86:617–639.

[10] Zender H, Mozos OM, Jensfelt P, Kruijff GJ, Burgard W. Conceptual spatial representations for indoor mobile robots. *Robot. Auton. Syst.* 2008, 56(6):493–502.

[11] Speer R, Chin J, Havasi C. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, San Francisco, USA, February 04–09, 2017, pp. 4444–4451.

[12] Pennington J, Socher R, Manning CD. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, Doha, Qatar, October 25–29, 2014, pp. 1532–1543.

[13] Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, *et al.* Attention is All you Need. In *Proceedings of the Thirty-First Advances in Neural Information Processing Systems*, Long Beach, USA, December 04–09, 2017, pp. 6000–6010.

[14] Gao C, Chen J, Liu S, Wang L, Zhang Q, *et al.* Room-and-object aware knowledge reasoning for remote embodied referring expression. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Nashville, USA, June 20–25, 2021, pp. 3064–3073.

[15] Qi Y, Wu Q, Anderson P, Wang X, Wang WY, *et al.* Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer*

Vision and Pattern Recognition, Seattle, USA, June 13–19, 2020, pp. 9982–9991.

[16] Long Y, Cai W, Wang H, Zhan G, Dong H. InstructNav: Zero-shot System for Generic Instruction Navigation in Unexplored Environment. *arXiv* 2024, ArXiv:2406.04882.

[17] Long Y, Li X, Cai W, Dong H. Discuss before moving: Visual language navigation via multi-expert discussions. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, Yokohama, Japan, May 13–17, 2024, pp. 17380–17387.

[18] Shah D, Equi MR, Osiński B, Xia F, Ichter B, *et al.* Navigation with large language models: Semantic guesswork as a heuristic for planning. In *Proceedings of The 7th Conference on Robot Learning*, Atlanta, USA, November 6–9, 2023, pp. 2683–2699.

[19] Huang C, Mees O, Zeng A, Burgard W. Visual language maps for robot navigation. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, London, UK, May 29–June 02, 2023, pp. 10608–10615.

[20] Huang W, Abbeel P, Pathak D, Mordatch I. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. In *International Conference on Machine Learning*, Baltimore, USA, July 17–23, 2022, pp. 9118–9147.

[21] Na S, Jeong G, Ahn BH, Young J, Krishna T, *et al.* Understanding Performance Implications of LLM Inference on CPUs. In *2024 IEEE International Symposium on Workload Characterization (IISWC)*, Vancouver, Canada, September 15–17, 2024, pp. 169–180.

[22] Stoelinga S. Calculating GPU memory for serving LLMs. Available: https://www.substratus.ai/blog/calculating-gpu-memory-for-llm (accessed on 07 February 2025).

[23] Elksnis A, Xue Z, Chen F, Wang N. Utility of Local Quantized Large Language Models in Semantic Navigation. In *2024 29th International Conference on Automation and Computing (ICAC)*, Sunderland, UK, August 28–30, 2024, pp. 1–6.

[24] Zhou J, Elksnis A, Fu Z, Chen B, Yang C. MultiMap3D: A Multi-Level Semantic Perceptual Map Construction Based on SLAM and Point Cloud Detection. In *2023 28th International Conference on Automation and Computing (ICAC)*, Birmingham, UK, 30 August–01 September, 2023, pp. 1–6.

[25] Chaplot DS, Gandhi DP, Gupta A, Salakhutdinov RR. Object goal navigation using goal-oriented semantic exploration. In *the 34th Annual Conference on Neural Information Processing Systems*. December 06–12, 2020, pp. 4247–4258.

[26] Huang W, Xia F, Xiao T, Chan H, Liang J, *et al.* Inner monologue: Embodied reasoning through planning with language models. *arXiv* 2022, ArXiv:2207.05608.

[27] Qiao Y, Liu Q, Liu J, Liu J, Wu Q. LLM as Copilot for Coarse-Grained Vision-and-Language Navigation. In *European Conference on Computer Vision*, Milan, Italy, September 29–October 04, 2024, pp. 459–476.

[28] Qiao Y, Lyu W, Wang H, Wang Z, Li Z, *et al.* Open-Nav: Exploring Zero-Shot Vision-and-Language Navigation in Continuous Environment with Open-Source LLMs. *arXiv* 2024, ArXiv:2409.18794.

[29] Deitke M, VanderBilt E, Herrasti A, Weihs L, Salvador J, *et al.* ProcTHOR: Large-Scale Embodied AI Using Procedural Generation. In *Proceedings of the 36th Advances in Neural Information*

Processing Systems, New Orleans, USA, December 06–12, 2022, pp. 5982–5994.

[30] Team G, Mesnard T, Hardin C, Dadashi R, Bhupatiraju S, *et al.* Gemma: Open Models Based on Gemini Research and Technology. *arXiv* 2024, ArXiv:2403.08295.

[31] Jiang AQ, Sablayrolles A, Mensch A, Bamford C, Chaplot DS, *et al.* Mistral 7B. *arXiv* 2023, ArXiv:2310.06825.

[32] AI@Meta. Llama 3 Model Card. Available: https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md (accessed on 18 April 2024).

[33] Kolve E, Mottaghi R, Han W, VanderBilt E, Weihs L, *et al.* Ai2-thor: An interactive 3d environment for visual ai. *arXiv* 2017, ArXiv:1712.05474.

[34] Poon J, Dryja T. The bitcoin lightning network: Scalable off-chain instant payments. 2016, pp. 1-59. Available: https://huggingface.co/vikhyatk/moondream2 (accessed on 09 September 2024).

[35] Team C. Chameleon: Mixed-Modal Early-Fusion Foundation Models. *arXiv* 2024, ArXiv:2405.09818.

[36] NVIDIA. Jetson AGX Xavier Developer Kit. Available: https://developer.nvidia.com/embedded/jetson-agx-xavier-developer-kit (accessed on 19 September 2024).

[37] Zheng K, Chitnis R, Sung Y, Konidaris G, Tellex S. Towards Optimal Correlational Object Search. In *IEEE International Conference on Robotics and Automation (ICRA)*, Philadelphia, USA, May 23–27, 2022, pp. 7313–7319.

[38] Gerganov G. GGUF. Available: https://github.com/ggml-org/ggml/blob/master/docs/gguf.md (accessed on 12 May 2025).

[39] Dettmers T, Pagnoni A, Holtzman A, Zettlemoyer L. Qlora: Efficient finetuning of quantized llms. In *the 37th Annual Conference on Neural Information Processing Systems*, New Orleans, USA, December 10–16, 2023 pp. 10088–10115.

[40] Wei J, Wang X, Schuurmans D, Bosma M, Xia F, *et al.* Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th Advances in Neural Information Processing Systems*, New Orleans, USA, November 28–December 09, 2022, pp. 24824–24837.

[41] Ollama. Available: https://ollama.com/ (accessed on 18 April 2024).