EfficientNav: Towards On-Device Object-Goal Navigation with Navigation Map Caching and Retrieval

Zebin Yang[1,2] **Sunjian Zheng**[3,4] **Tong Xie**[1,2] **Tianshi Xu**[1,2] **Bo Yu**[3†]
Fan Wang[3] **Jie Tang**[4] **Shaoshan Liu**[3] **Meng Li**[1,2,5†]

[1]Institute for Artificial Intelligence, Peking University
[2]School of Integrated Circuits, Peking University
[3]Shenzhen Institute of Artificial Intelligence and Robotics for Society
[4]School of Computer Science and Engineering, South China University of Technology
[5]Beijing Advanced Innovation Center for Integrated Circuits

Abstract

Object-goal navigation (ObjNav) tasks an agent with navigating to the location of a specific object in an unseen environment. Embodied agents equipped with large language models (LLMs) and online constructed navigation maps can perform ObjNav in a zero-shot manner. However, existing agents heavily rely on giant LLMs on the cloud, e.g., GPT-4, while directly switching to small LLMs, e.g., LLaMA3.2-11b, suffer from significant success rate drops due to limited model capacity for understanding complex navigation maps, which prevents deploying ObjNav on local devices. At the same time, the long prompt introduced by the navigation map description will cause high planning latency on local devices. In this paper, we propose EfficientNav to enable on-device efficient LLM-based zero-shot ObjNav. To help the smaller LLMs better understand the environment, we propose semantics-aware memory retrieval to prune redundant information in navigation maps. To reduce planning latency, we propose discrete memory caching and attention-based memory clustering to efficiently save and re-use the KV cache. Extensive experimental results demonstrate that EfficientNav achieves 11.1% improvement in success rate on HM3D benchmark over GPT-4-based baselines, and demonstrates 6.7× real-time latency reduction and 4.7× end-to-end latency reduction over GPT-4 planner. Our code is available on https://github.com/PKU-SEC-Lab/EfficientNav.

1 Introduction

Aiming to navigate an agent to an object specified by its category in an unknown environment, object-goal navigation (ObjNav) presents a crucial yet challenging task for embodied agents [1, 58, 43, 25]. To enhance the generality of ObjNav, recent studies have proposed leveraging the common-sense reasoning capabilities of large language models (LLMs) to achieve long-term planning in a zero-shot manner [40, 4, 54, 23, 65]. Specifically, LLMs act as planners and determine a series of navigation sub-goals to guide the agent to the final destination [11]. In such systems, navigation goals, maps of explored areas, and current observations are typically provided as context for the LLMs [10, 91, 8]. This information can be conceived as memory for LLMs, providing information of explored areas and visited objects to facilitate planning and decision-making.

[†]Corresponding author. Emails: boyu@cuhk.edu.cn, meng.li@pku.edu.cn

39th Conference on Neural Information Processing Systems (NeurIPS 2025).

 https://doi.org/10.52202/085713-0152

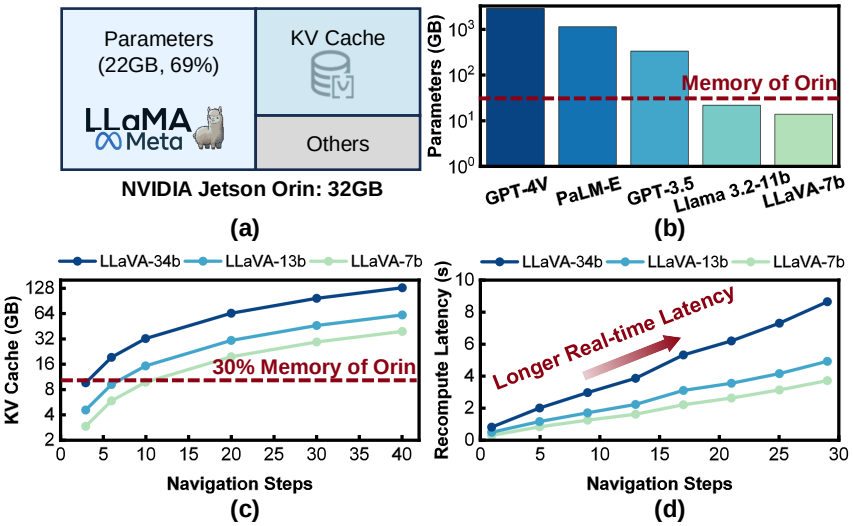


Figure 1: (a) Local devices (e.g., NVIDIA Jetson AGX Orin) exhibit constrained memory capacity (32GB). (b) Only smaller LLMs, such as Llama-3.2-11b, can be used as the planner due to memory limitations. (c) The KV cache of map information accumulates with navigation steps, exceeding the memory capacity. (d) Recomputing the KV cache incurs long real-time latency.

Though promising, LLM-based ObjNav systems face significant challenges due to the substantial memory requirements and computational complexity of LLMs. To ensure task success, giant LLMs, such as GPT-4, are typically employed, necessitating offloading these models to the cloud [63, 7, 91, 74, 66]. Such computing systems not only suffer from high communication latency, which degrades real-time performance [26, 53, 59, 44, 46, 36, 90], but also raise privacy concerns [18, 33, 52, 80, 16] and incur extremely high computational costs [5, 69]. Hence, a computing paradigm shift from offloading LLMs to the cloud towards accommodating LLMs on local servers or embedded devices, i.e., on-device computing, is required for ObjNav systems [24, 62].

However, enabling on-device ObjNav presents two primary design challenges, largely constrained by limited memory resources. First, only smaller LLMs can be deployed on local devices, thus causing planning performance drops. For example, as shown in Figure 1 (a) and (b), the NVIDIA Jetson AGX Orin [24] only has 32GB of DRAM, which can only accommodate smaller LLMs such as LLaMA-3.2-11b. Since larger LLMs typically have higher model capacity, enabling them to handle more complex and longer navigation tasks, on-device LLMs struggle to achieve the same level of performance as their cloud-based counterparts [8, 40]. Second, as context information accumulates during the navigation process, its corresponding KV-cache also grows and can eventually exceed the memory constraint (in Figure 1 (c), we set 30% of the Orin memory as the KV cache memory constraint, as the model weights also need to be stored in memory). And re-computing the KV-cache significantly slows down LLM decoding, introducing substantial compute latency (Figure 1 (d)).

In this paper, we propose EfficientNav, an on-device memory-augmented ObjNav system that enables efficient zero-shot in-door navigation. We observe that describing the navigation map can lead to long prompt lengths, and the corresponding KV-cache often cannot be fully stored due to memory constraints. Consequently, we select only a portion of the map description and load its KV cache for the LLM. However, the description selection will change the context order and the prefix of the corresponding KV cache. To enable reusing the retrieved KV cache despite the changing prefix, we propose discrete memory caching. This involves clustering the map information into groups and calculating the KV-cache for each group independently. Then, KV-caches within the same group are subject to cross-attention. To mitigate the impact of ignoring cross-attention between groups, we propose attention-based memory clustering. This technique uses LLM attention mechanisms to cluster related information into the same group. Furthermore, we observe that smaller LLMs may struggle to fully understand complex navigation maps. To address this, we propose semantics-aware memory retrieval to prune redundant map information during the group selection process, thereby improving the performance of smaller LLMs.

We summarize our contributions as follows: ❶ We propose discrete memory caching to prevent saving the KV cache of the whole map description and meet the memory constraints. ❷ We propose attention-based memory clustering to reduce the impact of ignoring cross-attention between groups. ❸ We propose semantics-aware memory retrieval to efficiently select the relevant groups and prune redundant map information. ❹ We conduct extensive experiments and show that EfficientNav can achieves 11.1% success rate improvements over GPT-4-based methods on HM3D dataset, and show 6.7× real-time latency and 4.7× end-to-end latency reduction over GPT-4 planner.

2 Related Work

LLM-based object goal navigation. According to the data processing pipeline, LLM-based ObjNav can be categorized into two paradigms: end-to-end and modular approaches. The end-to-end method, such as NaVid [83], directly converts RGB-D inputs into robot control policies, but incurs significant training costs [6, 27]. The modular approach, which is more prevalent due to simplicity, involves using LLMs as a high-level planner, and a lower-level controller handled by the robot itself. [40, 1, 54, 7, 23]. The LLM planner generates sub-goals to reach the final goal conditioned on the environment and instructions. The controller [56, 58], usually a small neural network, finds a trajectory from the current place to the sub-goal and controls the robot's motion in a real-time manner. We focus on optimizing the LLM planner, since it is the primary efficiency bottleneck due to its long inference latency and high memory cost [76, 10].

Memory Augmentation for object goal navigation. In zero-shot ObjNav tasks, the robot faces inefficiencies during sub-goal planning due to its limited field of view [7]. To achieve efficient environment exploration, mechanisms for memorizing navigation history are crucial for ObjNav [86, 87, 60, 88]. Maps built during navigation are typically leveraged to explicitly memorize the semantics and locations of visited areas and objects. While occupancy maps [40] can be used for memorization, graph-based navigation maps [10, 1, 91, 68] are increasingly favored for their ability to explicitly represent the semantics and locations of visited objects, which can be seamlessly integrated with LLM-based ObjNav [8, 47]. Hence, in this paper, we focus on the graph-based navigation maps.

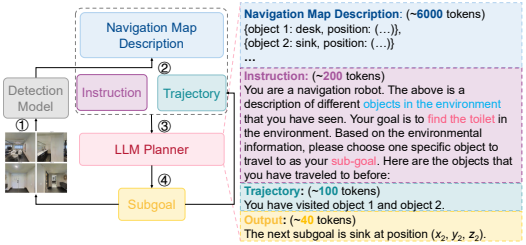


Figure 2: Software flow of LLM-based ObjNav.

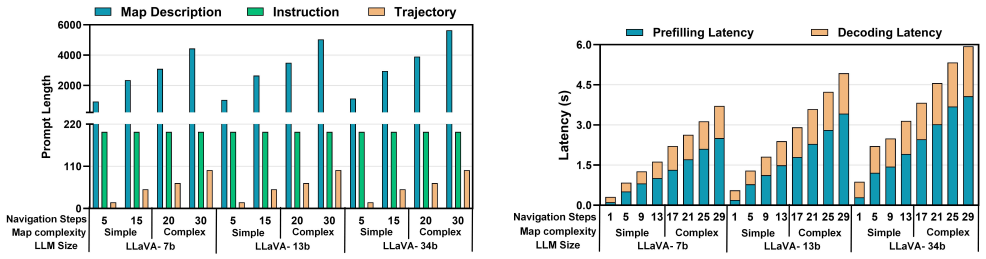
Efficient LLMs. Many approaches have been proposed to enable efficient LLM inference on devices with limited computational resources, including quantization [12, 34, 71, 28, 32, 17], pruning [41, 21, 14, 41, 45], knowledge distillation [35, 50, 73], etc. However, these methods are not specifically optimized for ObjNav. In particular, they do not address the trade-offs between LLM size and the accuracy of the task planner, nor the challenge of efficient KV-cache saving and retrieval as the navigated map expands in ObjNav scenarios. We will further discuss them in Section 3.1.

Comparisons with prior methods. Our approach employs a modular architecture that uses a graph-based navigation map to represent the semantic and spatial information of the navigated area. It leverages LLMs to reason based on this graph, current observations, and the goal to generate the next navigation target. While aligned with many prior architectural designs, our approach is distinguished by its efficient and highly accurate on-device ObjNav, as shown in Table 1. This is achieved through our efficient KV-cache retrieval and pruning mechanism. Notably, our method is orthogonal to and compatible with classic efficient LLM techniques such as quantization, knowledge distillation, etc.

Table 1: Comparison with prior methods.

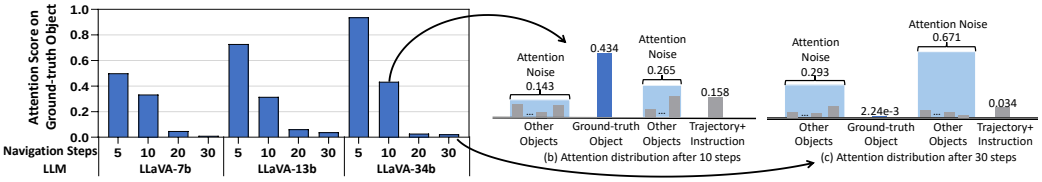
Method	Zero-shot	LLM	On-device Inference	Memory Augmented
ViKiNG [55]	✗	-	✓	✓
NaVid [83]	✗	Vicuna	✓	✗
Skip-SCAR [39]	✗	-	✓	✓
Pixel Navigation [7]	✓	GPT-4	✗	✗
InstructNav [40]	✓	GPT-4V	✗	✓
MapGPT [10]	✓	GPT-4	✗	✓
LFG [54]	✓	GPT-4	✗	✓
Imagine Before Go [88]	✓	GPT-4	✗	✓
SayNav [49]	✓	GPT-4	✗	✓
EfficientNav **(Ours)**	✓	LLaMA	✓	✓

https://doi.org/10.52202/085713-0152



(a) (b)

Figure 3: (a) Average length of different parts of prompts across navigation steps. Prompt lengths vary slightly across models due to divergent sub-goal selections influencing map complexity. (b) On-device LLM planning latency across navigation steps.



(a)

Figure 4: (a) Attention score on the most promising sub-goal (ground-truth object) in different navigation steps. Attention distribution after (b) 10 steps and (c) 30 steps using LLaVA-34b in sub-goal planning. As the amount of map information increases with the progression of navigation, the LLM can not fully understand the navigation map to focus on the most promising sub-goal. We observe similar problems in other small open-source LLMs such as LLaMA and Vicuna. The ground-truth object is chosen by GPT-4o.

3 EfficientNav: Memory Augmented On-device Navigation System

3.1 Challenges and Overview

ObjNav software pipeline. In our ObjNav, the goal-finding process is composed of iterative sub-goal finding tasks until the final goal is reached. Figure 2 illustrates our ObjNav's navigation pipeline for finding sub-goals and the ultimate goal. The first step involves generating semantic and distance information from RGB-D sensor captures using detection models, e.g., Grounding Dino [37]. Next, this semantic and spatial information is organized into the graph-based navigation map (we show an example in Appendix B). Third, the updated map and goal instructions are provided to the LLM for goal planning. Finally, if the target object is already present in the map, we choose it as the next goal; otherwise, the planner chooses a promising object in the map as the next sub-goal and continues detecting new environmental information after reaching the sub-goal. The computation challenges of this method lies in efficiently and accurately running the LLM as the map expands, which we discuss as follows.

Challenge 1: tight memory constraints of local device limit the saving of KV cache of the navigation map description. As discussed in Section 2 and shown in Figure 3 (a), with the progression of navigation, the amount of map information rapidly increases, thus introducing long prompt length, usually up to thousands of tokens. Re-computing the KV cache in each planning process will cause a long prefilling time, which is shown in Figure 3 (b). To avoid this cost, [22] saves the KV cache of history information in server memory, while as shown in Figure 1 (c), this can not meet the memory constraints of local devices. So a new memory caching mechanism is needed.

Challenge 2: the tight memory constraint forces to use smaller LLMs, which have poorer model capacity and cause success rate degradation. To meet the memory constraints of local devices, usually tens of gigabytes, we need to use smaller LLMs, e.g., LLaVA-7b. However, directly using them to replace giant LLMs, such as GPT-4, will cause a significant success rate drop [8, 29]. In practice, we find this comes from that the smaller LLM with lower capacity can not fully understand complex navigation maps. With the progression of navigation, the LLM can not pay attention to the

https://doi.org/10.52202/085713-0152

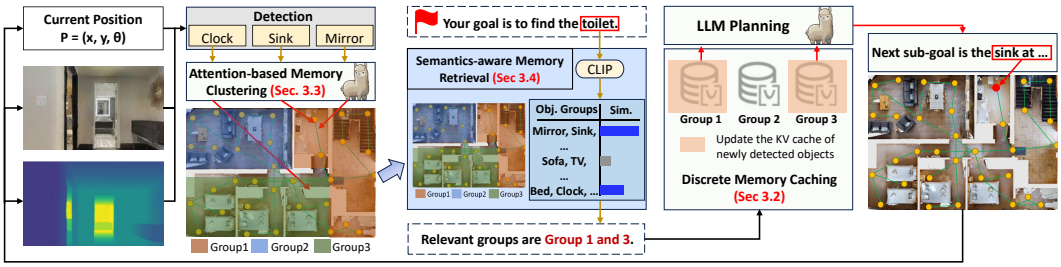
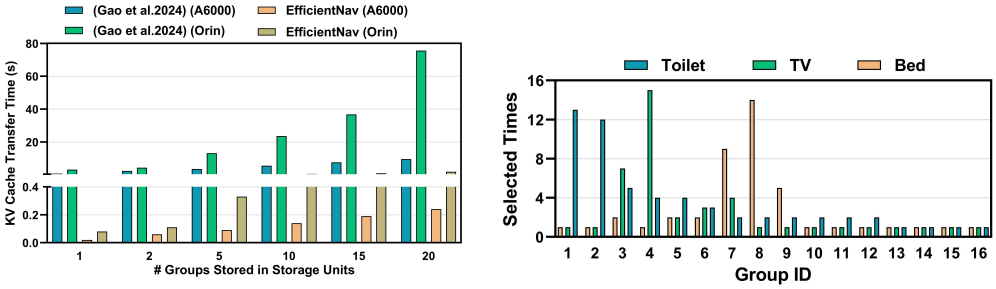


Figure 5: Overview of EfficientNav. Sim. stands for similarity.



(a) (b)

Figure 6: (a) Memory transfer time of LLaVA-7b when using low-speed storage units. (b) For different final goals, the selected times of different groups are different, thus showing different relevance. For each final goal, the navigation task is conducted 15 times with different starting points. The groups are selected by GPT-4o.

important information among the huge amount of map information and choose the correct sub-goal, which is shown in Figure 4. So a memory selection strategy is needed to remove the redundant information for the smaller LLM.

EfficientNav Overview. Based on these observations, we propose EfficientNav to enable efficient zero-shot ObjNav on local devices. Figure 5 demonstrates the overview of EfficientNav. To meet the memory constraints, we propose discrete memory caching to cluster objects into groups and calculate the KV cache of each group individually. Then we only select a portion of groups to LLM and load their KV cache to device memory. To accurately cluster information and improve success rate, we propose attention-based memory clustering, using LLM attention to guide group clustering. To help the smaller LLMs better understand the navigation map, in the group selection process, we propose semantic-aware memory retrieval to efficiently prune redundant map information.

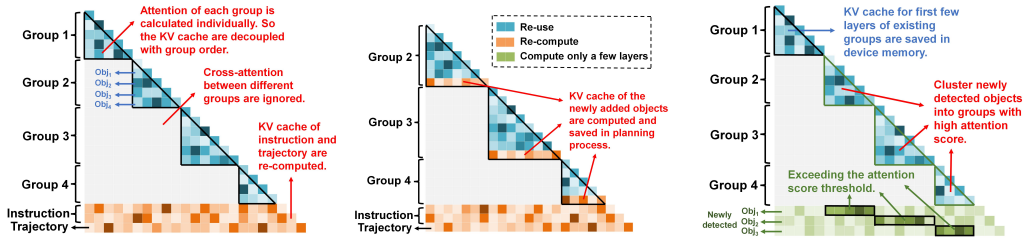
3.2 Discrete Memory Caching

As discussed in Section 3.1, the memory constraints of local devices prevent full storage of the KV cache of navigation map descriptions. To mitigate this, [13] offloads the KV cache in high-capacity but low-speed storage units, e.g., CPU host memory for NVIDIA RTX A6000 or disk for Jetson Orin, and loads the KV cache on-demand during decoding. However, since local memory can not retain all the KV cache, this method requires repeated loading of the KV cache to device memory in each decoding phase. However, in each planning process, LLM needs to decode multiple tokens (around 40 in practice). The frequent communication between device memory and storage units will also cause huge latency, which is shown in Figure 6 (a).

To meet the memory constraint while avoiding re-computation and frequent memory transfers, we propose a discrete memory caching strategy. Building on [13], we store the KV cache in low-speed storage units and load the required KV cache into device memory, but introduce a critical optimization. As the navigation map contains redundancy, we only select partial important information according to the memory budget and load the KV cache only once, thus reducing memory transfer cost.

However, when selecting information, the order of context in the prompt will change. When pre-calculating the KV cache, existing works [38, 89] calculate the full attention of the whole context. In this approach, only the KV cache of its longest common prefix with the selected information can be

https://doi.org/10.52202/085713-0152



(a) (b) (c)

Figure 7: (a) When using discrete memory caching, the KV cache of existing groups can be reused in LLM planning. (b) When groups with newly added objects are selected during LLM planning process, the KV cache of these objects can be directly calculated and saved without impacting existing KV cache. (c) When using attention-based memory clustering, the newly detected objects will be clustered into different groups according to attention distribution.

reused. The KV cache after the changed position needs to be recomputed. To address this, we cluster objects in the navigation map into groups and compute KV cache of each group individually, as shown in Figure 7 (a). Then, only partial groups are selected to prompt the LLM planner. Hence, we can decouple the group KV cache and the group order, thus making full usage of the saved KV cache and avoiding re-computation. As shown in Figure 5, in each navigation step, newly detected objects will be added to existing groups, and we need to add their KV cache to the group's KV cache. To avoid changing context order within the group and impacting the existing KV cache, we concatenate the information of new objects at the end of the group information. As shown in Figure 7 (b), in the planning process, when groups with newly added objects are selected, the KV cache of these objects can be directly calculated and saved, without introducing extra computation. We follow the position encoding strategy in LLM inference as [15].

However, discrete memory caching will cause the ignorance of cross-attention between groups, leading to LLM performance drop [75, 19]. We also need to ensure that the groups with important information are not removed in group selection. Therefore, a correct memory clustering and memory selection mechanism is very important. We will explain our design in Section 3.3 and 3.4.

3.3 Attention-based Memory Clustering

As discussed in Section 3.2, to reduce the impact of ignoring cross-attention and maintain a high success rate, we need to cluster the information accurately and efficiently. Existing works [22, 75] partition context into uniform-length chunks, neglecting the relationships between objects in the navigation map. For example, the relationship between the oven and the pot is closer than the relationship between the oven and the bed. By grouping objects with closer relationships together and calculating their cross-attention, LLMs can better understand the navigation map and abstract the environment of a larger area. Meanwhile, the grouping granularity is also important. In contrast, if the granularity is too coarse, the KV cache size of each group will become large. Hence, only a few groups can be selected for the LLM, which will enhance the difficulty of selection and may neglect important information.

To adaptively control object clustering and group granularity, we use the attention of LLM itself to cluster the newly detected objects into different groups. As shown in Figure 7 (c), in the clustering process, we input the information of existing groups and the newly detected objects into LLM, and only infer a few layers (in practice, we find around $\frac{1}{10}$ layers of the whole model is enough). If the average attention between a newly detected object and an existing group exceeds a specific threshold, we cluster this object into the group. Otherwise, the remaining objects will be added to a new group. For the first group, we simply cluster the detected objects in the first navigation step into a group.

By using attention-based memory clustering, we can reduce the difference between discrete attention and full attention. Note that the clustering process will not introduce much extra computation and latency. This is because we only compute the first few layers of LLM, and the KV cache of existing groups has been pre-computed and its cache of the first few layers are kept in device memory.

https://doi.org/10.52202/085713-0152

3.4 Semantics-aware Memory Retrieval

As discussed in Section 3.1, the smaller LLM cannot fully understand complex navigation maps. Thus, a memory selection mechanism is needed to remove redundant information. [54] empirically selects environmental information based on object positions. However, we observe that when the final goal changes, the relevance of each group to the final goal also changes, as shown in Figure 6 (b). This empirical selection can not adapt to different final goals, thus showing lower performance. At the same time, it does not consider different device memory budgets. [8] uses LLM to select relevant information, while this will introduce extra LLM calls and long real-time latency.

Based on this, we propose semantics-aware memory retrieval, efficiently selecting groups according to semantic information to adapt to different sub-goals. As the task of removing redundant information is much easier than finding a specific sub-goal, to improve efficiency, we conduct group selection and sub-goal planning in a small-large model collaboration mode. For easier group selection, we use CLIP model [11] instead of LLM, which only has around 100M parameters and lower inference latency. For the harder sub-goal planning, we use LLM to select one specific object as the sub-goal. In the information selection process, we use CLIP to encode the object information in each group and the final goal. We then calculate the similarity between the encoding results of the final goal and each group, which is viewed as the probability for a group to include potential sub-goals. We consider the group relevant to the final goal only if the probability is larger than a threshold. To adapt to different devices with different memory budgets, we formulate the group selection as a knapsack problem:

$$\text{maximize} \quad P = \sum_{i=1}^{n} (P_i - threshold) \cdot x_i \tag{1}$$

$$\text{subject to} \quad \sum_{i=1}^{n} M_i \cdot x_i \leq M, \;\; x_i \in \{0, 1\} \tag{2}$$

where P_i denotes the probability to include sub-goals in group i, M_i and M denotes the memory usage of KV cache of group i and device memory budget, x_i denotes whether to select group i, n denotes the number of groups. After solving the knapsack problem, we can select truly relevant information for LLM and meet the memory budget at the same time.

By using semantics-aware memory retrieval, we can efficiently select relevant groups adapting to different final goals and devices. Note that in each planning process, we only update the decoding results of the groups with newly detected objects. As the inference time of CLIP is much lower than LLM, the latency of the selection process is negligible. At the same time, between two adjacent planning processes, the navigation map will not change much. So it is likely that some group are both selected in the current step and last step. Hence, their KV cache has been loaded into device memory, thus reducing the KV cache loading time, which we will further show in our experiments.

4 Experiments

4.1 Experiment Setup

We evaluate EfficientNav on the HM3D dataset [3] based on the Habitat simulation platform [61]. In the simulation platform, the robot can access RGBD observation of the environment. In each task, the robot is placed at a different starting point in the environment and is only instructed to find a specific object, e.g., “TV”, “chair”, “sofa”, "bed", "toilet", "plant" etc., which is harder than tasks giving detailed, step-by-step directions [2, 48]. For accuracy, we report two metrics: i) the average success rate (SR), and ii) the success rate penalized by path length (SPL), which both evaluates the accuracy and the efficiency of robot trajectory. For system efficiency, we report two metrics: i) real-time latency (RtL): the average robot planning time

Table 2: SR and SPL comparison.

Method	Zero-shot	LLM	SR	SPL
DD-PPO [67]	✗	-	27.9	14.2
SemExp [9]	✗	-	37.9	18.8
Habitat-web [51]	✗	-	41.5	16.0
OVRL [72]	✗	-	62.0	26.8
ZSON [42]	✓	-	25.5	12.6
PixelNav [7]	✓	GPT-4	37.9	20.5
ESC [92]	✓	-	39.2	22.3
VoroNav [68]	✓	GPT-3.5	42.0	26.0
LLaVA Planner-34b [10]	✓	LLaVA-34b	42.7	21.0
L3MVN [79]	✓	RoBERTa-large	50.4	23.1
InstructNav [40]	✓	GPT-4V	58.0	20.9
LFG [54]	✓	GPT-4	68.9	36.0
EfficientNav-11b	✓	LLaMA3.2-11b	**74.2**	**39.5**
EfficientNav-34b	✓	LLaVA-34b	**80.0**	**41.5**

https://doi.org/10.52202/085713-0152

in one navigation step. ii) End-to-end latency (E2EL): the average latency for the robot to complete one navigation task (including multiple navigation steps and robot moving time). We implement our methods on 4 LLMs, LLaVA-7b, LLaVA-13b, LLaVA-34b, and LLaMA3.2-11b on NVIDIA A6000 GPU and Jetson Orin. We show the network architecture and implementation details in Appendix D and G. We also show an example of EfficientNav pipeline in Section B and C.

4.2 Main Results

Comparison with state-of-the-art. The comparison of SR and SPL is shown in Table 2. Compared with learning-based methods, EfficienNav achieves 18.0% SR and 14.7% SPL improvements over the prior-art method OVRL [72], without any training cost. This demonstrates the advantage of LLM-based navigation system, as discussed in Section 1. Compared with zero-shot methods, EfficientNav achieves 11.1% SR and 5.5% SPL improvements over LFG [54], which utilizes GPT-4. Compared with naive LLaVA-34b planner [10], EfficientNav achieves 37.3% SR and 20.5% SPL improvements. This is because we use semantics-aware memory retrieval, which helps the LLM focus on the most relevant groups and removes redundant information.

The latency comparison is shown in Table 3. Compared with GPT-4 planner [10], EfficientNav achieves 6.7× real-time latency reduction and 4.7× end-to-end latency reduction, by saving the communication time to the cloud server. Compared with naive LLaVA planner, EfficcientNav achieves 8.8× and 6.5× real-time latency reduction and 3.7× and 4.4× end-to-end latency reduction on LLaMA3.2-11b and LLaVA-34b. This mainly comes from using discrete memory caching to avoid re-computation in the prefilling stage of planning. Prior-art solutions such as vllm can accelerate LLM inference, but they can not solve the problem of high re-computation cost, which is the main bottleneck. Compared with vllm, EfficientNav achieves 6.5× and 5.1× real-time latency reduction and 3.1× and 3.8× end-to-end latency reduction on LLaMA3.2-11b and LLaVA-34b.

Table 3: Average latency comparison on A6000.

Method	LLM	RtL	E2EL
GPT-4 Planner [10]	GPT-4	5.80s	59.34s
LLaMA Planner-11b [10]	LLaMA3.2-11b	3.07s	46.40s
vllm [30]	LLaMA3.2-11b	2.27s	39.78s
EfficientNav-11b (**Ours**)	LLaMA3.2-11b	0.35s	12.70s
LLaVA Planner-34b [10]	LLaVA-34b	5.63s	55.32s
vllm [30]	LLaVA-34b	4.43s	47.95s
EfficientNav-34b (**Ours**)	LLaVA-34b	0.87s	12.51s

Real-time latency in different navigation steps. The amount of map information increases in each navigation step. As shown in Figure 8, when using traditional serving methods [30], the increasing prompt length will introduce high recomputation cost and increasing real-time latency. However, when using EfficientNav, the real-time latency stabilizes after a certain navigation step, as we use semantics-aware memory retrieval to select partial groups for LLM according to memory budget. With the same amount of information given to LLM, EffcientNav also shows lower real-time latency, as discrete memory caching saves the re-computation time of LLM prefilling.

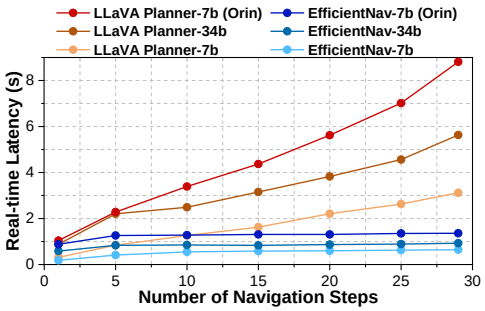


Figure 8: Comparison of real-time latency in different navigation steps on A6000 and Jetson Orin.

Planning latency breakdown. Figure 9 shows the latency breakdown of one navigation step. Compared to traditional serving methods [30], by avoiding re-computation, discrete memory caching can significantly reduce the prefilling time of planning by around 20×. And semantics-aware memory retrieval removes the redundant information, thus reducing the prompt length and reducing the computation cost and latency of the decoding stage of planning. The latency of memory clustering and memory selection is negligible, as discussed in Section 3.3 and 3.4.

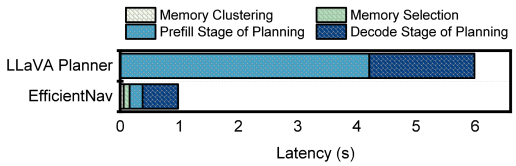


Figure 9: Planning latency (s) breakdown for LLaVA planner and EfficientNav on A6000.

https://doi.org/10.52202/085713-0152

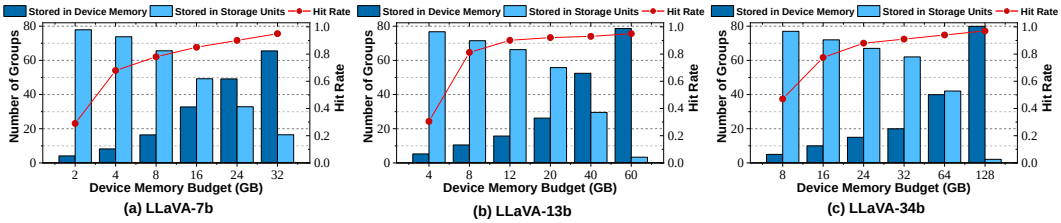


Figure 10: Comparison of memory caching distribution and memory cache hit rate across different device memory budgets for KV cache, using (a) LLaVA-7b, (b) LLaVA-13b, and (c) LLaVA-34b. Cache hit rate is defined as the proportion of selected groups that have been stored in device memory.

Memory caching distribution and cache hit rate of EfficientNav. When using EfficientNav, in each planning process, we only select relevant groups for LLM. With larger device memory budgets for KV cache, we can store more groups in device memory. As discussed in Section 3.4, some selected groups may be just stored in device memory, so we do not need to reload these groups from storage units. We define the proportion of these groups as the cache hit rate. A high hit rate will lead to a low caching loading time. As shown in Figure 10, the cache hit rate of EfficientNav increases with the device memory budget and rapidly reaches a high level. This is because when the memory budget increases, more relevant groups are stored in device memory. At the same time, the semantics of each group will not change much between adjacent planning processes. So it is likely that some relevant groups are just loaded to device memory in the last few steps, leading to a high cache hit rate.

4.3 Ablation Study

Individual influence of our methods. Table 4 shows the individual influence of our proposed methods. In naive LLaVA planner, we empirically cluster and select environmental information according to object positions [85, 54]. By using discrete memory caching, compared with naive LLaVA planner [10], we achieve 2.3× real-time latency and 1.5× end-to-end latency reduction, by re-using KV cache of navigation map description and avoiding re-computation. After attention-based memory clustering is used, objects in each selected group have high correlation, which helps the LLM to better understand the environment and reduces the impact of ignoring cross-attention at the same time, thus improving the success rate. When using semantics-aware memory retrieval, we can select the most related groups, which further improves the success rate. We also achieve 2.7× real-time latency and 2.6× end-to-end latency reduction, by selecting groups according to memory budget and avoiding frequent communication with storage units.

Table 4: Ablation study on EfficientNav methods with LLaVA-34b.

Method	SR	SPL	RtL	E2EL
LLaVA Planner	42.7	21.0	5.63s	55.32s
+Discrete Memory Caching	43.1	21.0	2.42s	36.94s
+Attention-based Memory Clustering	63.3	34.2	2.32s	32.58s
+Semantics-aware Memory Retrieval	80.0	41.5	0.87s	12.51s

Impact of device memory budget for KV cache. In semantics-aware memory retrieval, we select relevant groups according to the device memory budget. Here we evaluate the impact of memory budget on success rate and latency using LLaVA-34b. As shown in Table 5, on most cases, our method shows good robustness. However, when the memory budget for KV cache is too small, fewer relevant groups can be selected. Hence, the potential sub-goal may be ignored, which causes success rate drops and longer end-to-end latency. For a larger memory budget, more relevant groups can be selected. While the longer prompt length may cause longer planning time, which will slightly increase the real-time latency.

Table 5: Ablation study on different device memory budgets for KV cache with LLaVA-34b.

Memory Budget	SR	SPL	RtL	E2EL
16GB	74.7	37.3	0.59s	14.24s
24GB	79.0	39.1	0.71s	14.29s
32GB	80.0	41.5	0.87s	12.51s
40GB	80.3	41.9	0.93s	11.72s

Comparison with sparse attention approaches. Our discrete memory caching is similar to adaptive sparse attention methods [20, 31, 84]. However, the goals of general adaptive sparse attention and EfficientNav are different: EfficientNav clusters objects into different groups and calculates

https://doi.org/10.52202/085713-0152

the attention of each group individually. This structured sparse approach can decouple the group order and KV cache computation of each group, thus enabling KV reuse when we retrieve different groups in navigation planning and reducing real-time latency. However, for general adaptive sparse attention methods, their purpose is to minimize the difference between sparse attention and full attention and accelerate the attention calculation. The comparison with sparse attention approaches is shown in Table 6. Just using adaptive sparse attention methods shows huge accuracy drops. This is because these methods minimize the difference between sparse attention and full attention. But without memory retrieval, even full attention shows a low success rate, as discussed in Section 3.1. At the same time, without discrete memory caching, these methods need to recompute the KV cache of the navigation map description because of map changes, thus showing longer real-time latency.

Table 6: Comparison with sparse attention approaches with LLaVA-34b.

Method	SR	SPL	Real-time latency
Minference [20]	35.3	18.0	3.88s
FlexPrefill [31]	36.7	20.1	3.02s
EfficientNav	80.0	41.5	0.87s

5 Conclusion and Limitations

This work proposes EfficientNav, which enables efficient zero-shot ObjNav on local devices. We propose discrete memory caching to enable re-using KV cache and avoid re-computation in LLM planning. We also propose attention-based memory clustering to reduce the impact of ignoring cross-attention. To improve planning performance of smaller LLMs, we propose semantics-aware memory retrieval to remove the redundancy in navigation map. EfficientNav overcomes all existing baselines and enables running zero-shot ObjNav on local devices. While our study shows promising results, it has some limitations. We mainly focus on LLM-based navigation in this paper, while the inference speed of LLM can not reach that of small models even after acceleration. And EfficientNav should be carefully used in applications that need extremely low real-time latency.

6 Acknowledgments

This work was supported in part by NSFC under Grant 62495102, Grant 92464104, and Grant 62341407, in part by the National Key Research and Development Program under Grant 2024YFB4505004, in part by Beijing Municipal Science and Technology Program under Grant Z241100004224015, in part by 111 Project under Grant B18001, and in part by Longgang District Shenzhen's "Ten Action Plan" for Supporting Innovation Projects under Grant LGKCSDPT2024003 and LGKCSDPT2024004.

References

[1] Dong An, Hanqing Wang, Wenguan Wang, Zun Wang, Yan Huang, Keji He, and Liang Wang. Etpnav: Evolving topological planning for vision-language navigation in continuous environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[2] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and Anton Van Den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3674–3683, 2018.

[3] Szot Andrew, Yadav Karmesh, Clegg Alex, Berges Vincent-Pierre, Gokaslan Aaron, Chang Angel, Savva Manolis, Kira Zsolt, and Batra Dhruv. Habitat rearrangement challenge 2022. `https://aihabitat.org/challenge/2022_rearrange`, 2022.

[4] Abrar Anwar, John Welsh, Joydeep Biswas, Soha Pouya, and Yan Chang. Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation. *arXiv preprint arXiv:2409.13682*, 2024.

[5] Aram Bahrini, Mohammadsadra Khamoshifar, Hossein Abbasimehr, Robert J Riggs, Maryam Esmaeili, Rastin Mastali Majdabadkohne, and Morteza Pasehvar. Chatgpt: Applications, opportunities, and threats. In *2023 Systems and Information Engineering Design Symposium (SIEDS)*, pages 274–279. IEEE, 2023.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[7] Wenzhe Cai, Siyuan Huang, Guangran Cheng, Yuxing Long, Peng Gao, Changyin Sun, and Hao Dong. Bridging zero-shot object navigation and foundation models through pixel-guided navigation skill. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5228–5234. IEEE, 2024.

[8] Yihan Cao, Jiazhao Zhang, Zhinan Yu, Shuzhen Liu, Zheng Qin, Qin Zou, Bo Du, and Kai Xu. Cognav: Cognitive process modeling for object goal navigation with llms. *arXiv preprint arXiv:2412.10439*, 2024.

[9] Devendra Singh Chaplot, Dhiraj Prakashchand Gandhi, Abhinav Gupta, and Russ R Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. *Advances in Neural Information Processing Systems*, 33:4247–4258, 2020.

[10] Jiaqi Chen, Bingqian Lin, Ran Xu, Zhenhua Chai, Xiaodan Liang, and Kwan-Yee Wong. Mapgpt: Map-guided prompting with adaptive path planning for vision-and-language navigation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9796–9810, 2024.

[11] Vishnu Sashank Dorbala, Gunnar Sigurdsson, Robinson Piramuthu, Jesse Thomason, and Gaurav S Sukhatme. Clip-nav: Using clip for zero-shot vision-and-language navigation. *arXiv preprint arXiv:2211.16649*, 2022.

[12] Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*, 2022.

[13] Bin Gao, Zhuomin He, Puru Sharma, Qingxuan Kang, Djordje Jevdjic, Junbo Deng, Xingkun Yang, Zhou Yu, and Pengfei Zuo. {Cost-Efficient} large language model serving for multi-turn conversations with {CachedAttention}. In *2024 USENIX Annual Technical Conference (USENIX ATC 24)*, pages 111–126, 2024.

[14] Shangqian Gao, Chi-Heng Lin, Ting Hua, Zheng Tang, Yilin Shen, Hongxia Jin, and Yen-Chang Hsu. Disp-llm: Dimension-independent structural pruning for large language models. *Advances in Neural Information Processing Systems*, 37:72219–72244, 2024.

[15] In Gim, Guojun Chen, Seung-seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. Prompt cache: Modular attention reuse for low-latency inference. *Proceedings of Machine Learning and Systems*, 6:325–338, 2024.

[16] Meng Hao, Hongwei Li, Hanxiao Chen, Pengzhi Xing, Guowen Xu, and Tianwei Zhang. Iron: Private inference on transformers. *Advances in neural information processing systems*, 35:15718–15731, 2022.

[17] Coleman Hooper, Sehoon Kim, Hiva Mohammadzadeh, Michael W Mahoney, Sophia Shao, Kurt Keutzer, and Amir Gholami. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *Advances in Neural Information Processing Systems*, 37:1270–1303, 2024.

[18] Xiaoyang Hou, Jian Liu, Jingyu Li, Yuhan Li, Wen-jie Lu, Cheng Hong, and Kui Ren. Ciphergpt: Secure two-party gpt inference. *Cryptology ePrint Archive*, 2023.

[19] Junhao Hu, Wenrui Huang, Haoyi Wang, Weidong Wang, Tiancheng Hu, Qin Zhang, Hao Feng, Xusheng Chen, Yizhou Shan, and Tao Xie. Epic: Efficient position-independent context caching for serving large language models. *arXiv preprint arXiv:2410.15332*, 2024.

[20] Huiqiang Jiang, Yucheng Li, Chengruidong Zhang, Qianhui Wu, Xufang Luo, Surin Ahn, Zhenhua Han, Amir H Abdi, Dongsheng Li, Chin-Yew Lin, et al. Minference 1.0: Accelerating pre-filling for long-context llms via dynamic sparse attention. *Advances in Neural Information Processing Systems*, 37:52481–52515, 2024.

 https://doi.org/10.52202/085713-0152

[21] Yikun Jiang, Huanyu Wang, Lei Xie, Hanbin Zhao, Hui Qian, John Lui, et al. D-llm: A token adaptive computing resource allocation strategy for large language models. *Advances in Neural Information Processing Systems*, 37:1725–1749, 2024.

[22] Chao Jin, Zili Zhang, Xuanlin Jiang, Fangyue Liu, Xin Liu, Xuanzhe Liu, and Xin Jin. Ragcache: Efficient knowledge caching for retrieval-augmented generation. *arXiv preprint arXiv:2404.12457*, 2024.

[23] Zhao Kaichen, Song Yaoxian, Zhao Haiquan, Liu Haoyu, Li Tiefeng, and Li Zhixu. Towards coarse-grained visual language navigation task planning enhanced by event knowledge graph. *arXiv preprint arXiv:2408.02535*, 2024.

[24] Leela S Karumbunathan. Nvidia jetson agx orin series, 2022.

[25] Serge Kernbach and Kristof Jebens. Development of a cost-efficient autonomous mav for an unstructured indoor environment. *arXiv preprint arXiv:1111.0500*, 2011.

[26] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[27] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, et al. Openvla: An open-source vision-language-action model. *arXiv preprint arXiv:2406.09246*, 2024.

[28] Sehoon Kim, Coleman Hooper, Amir Gholami, Zhen Dong, Xiuyu Li, Sheng Shen, Michael W Mahoney, and Kurt Keutzer. Squeezellm: Dense-and-sparse quantization. *arXiv preprint arXiv:2306.07629*, 2023.

[29] Taewoong Kim, Byeonghwi Kim, and Jonghyun Choi. Multi-modal grounded planning and efficient replanning for learning embodied agents with a few examples. *arXiv preprint arXiv:2412.17288*, 2024.

[30] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pages 611–626, 2023.

[31] Xunhao Lai, Jianqiao Lu, Yao Luo, Yiyuan Ma, and Xun Zhou. Flexprefill: A context-aware sparse attention mechanism for efficient long-sequence inference. *arXiv preprint arXiv:2502.20766*, 2025.

[32] Shiyao Li, Xuefei Ning, Ke Hong, Tengxuan Liu, Luning Wang, Xiuhong Li, Kai Zhong, Guohao Dai, Huazhong Yang, and Yu Wang. Llm-mq: Mixed-precision quantization for efficient llm deployment. In *NeurIPS 2023 Efficient Natural Language and Speech Processing Workshop*, pages 1–5, 2023.

[33] Chenqi Lin, Tianshi Xu, Zebin Yang, Runsheng Wang, Ru Huang, and Meng Li. Fastquery: Communication-efficient embedding table query for private llm inference. *arXiv preprint arXiv:2405.16241*, 2024.

[34] Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Wei-Ming Chen, Wei-Chen Wang, Guangxuan Xiao, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for on-device llm compression and acceleration. *Proceedings of Machine Learning and Systems*, 6:87–100, 2024.

[35] Jiaheng Liu, Chenchen Zhang, Jinyang Guo, Yuanxing Zhang, Haoran Que, Ken Deng, Jie Liu, Ge Zhang, Yanan Wu, Congnan Liu, et al. Ddk: Distilling domain knowledge for efficient large language models. *Advances in Neural Information Processing Systems*, 37:98297–98319, 2024.

[36] Runze Liu, Jianlei Yang, Yiran Chen, and Weisheng Zhao. eslam: An energy-efficient accelerator for real-time orb-slam on fpga platform. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[37] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Qing Jiang, Chunyuan Li, Jianwei Yang, Hang Su, et al. Grounding dino: Marrying dino with grounded pre-training for open-set object detection. In *European conference on computer vision*, pages 38–55. Springer, 2024.

[38] Shu Liu, Asim Biswal, Audrey Cheng, Xiangxi Mo, Shiyi Cao, Joseph E Gonzalez, Ion Stoica, and Matei Zaharia. Optimizing llm queries in relational workloads. *arXiv preprint arXiv:2403.05821*, 2024.

[39] Yaotian Liu and Jeff Zhang. Skip-scar: A modular approach to objectgoal navigation with sparsity and adaptive skips. *arXiv preprint arXiv:2405.14154*, 2024.

[40] Yuxing Long, Wenzhe Cai, Hongcheng Wang, Guanqi Zhan, and Hao Dong. Instructnav: Zero-shot system for generic instruction navigation in unexplored environment. *arXiv preprint arXiv:2406.04882*, 2024.

[41] Xinyin Ma, Gongfan Fang, and Xinchao Wang. Llm-pruner: On the structural pruning of large language models. *Advances in neural information processing systems*, 36:21702–21720, 2023.

[42] Arjun Majumdar, Gunjan Aggarwal, Bhavika Devnani, Judy Hoffman, and Dhruv Batra. Zson: Zero-shot object-goal navigation using multimodal goal embeddings. *Advances in Neural Information Processing Systems*, 35:32340–32352, 2022.

[43] Ali Reza Manashty, Amir Rajabzadeh, and Zahra Forootan Jahromi. A scenario-based mobile application for robot-assisted smart digital homes. *arXiv preprint arXiv:1009.5398*, 2010.

[44] Yongguo Mei, Yung-Hsiang Lu, Yu Charlie Hu, and CS George Lee. Deployment of mobile robots with energy and timing constraints. *IEEE Transactions on robotics*, 22(3):507–522, 2006.

[45] Saurav Muralidharan, Sharath Turuvekere Sreenivas, Raviraj Joshi, Marcin Chochowski, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro, Jan Kautz, and Pavlo Molchanov. Compact language models via pruning and knowledge distillation. *Advances in Neural Information Processing Systems*, 37:41076–41102, 2024.

[46] Sean Murray, William Floyd-Jones, Ying Qi, George Konidaris, and Daniel J Sorin. The microarchitecture of a real-time robot motion planning accelerator. In *2016 49th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO)*, pages 1–12. IEEE, 2016.

[47] Dujun Nie, Xianda Guo, Yiqun Duan, Ruijun Zhang, and Long Chen. Wmnav: Integrating vision-language models into world models for object goal navigation. *arXiv preprint arXiv:2503.02247*, 2025.

[48] Yuankai Qi, Qi Wu, Peter Anderson, Xin Wang, William Yang Wang, Chunhua Shen, and Anton van den Hengel. Reverie: Remote embodied visual referring expression in real indoor environments. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9982–9991, 2020.

[49] Abhinav Rajvanshi, Karan Sikka, Xiao Lin, Bhoram Lee, Han-Pang Chiu, and Alvaro Velasquez. Saynav: Grounding large language models for dynamic planning to navigation in new environments. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 34, pages 464–474, 2024.

[50] Mrigank Raman, Pranav Mani, Davis Liang, and Zachary Lipton. For distillation, tokens are not all you need. In *NeurIPS 2023 Workshop on Instruction Tuning and Instruction Following*, 2023.

[51] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5173–5183, 2022.

[52] Xuanle Ren, Zhaohui Chen, Zhen Gu, Yanheng Lu, Ruiguang Zhong, Wen-Jie Lu, Jiansong Zhang, Yichi Zhang, Hanghang Wu, Xiaofu Zheng, et al. Cham: A customized homomorphic encryption accelerator for fast matrix-vector product. In *2023 60th ACM/IEEE Design Automation Conference (DAC)*, pages 1–6. IEEE, 2023.

 https://doi.org/10.52202/085713-0152

[53] Deval Shah, Ningfeng Yang, and Tor M Aamodt. Energy-efficient realtime motion planning. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, pages 1–17, 2023.

[54] Dhruv Shah, Michael Robert Equi, Błażej Osiński, Fei Xia, Brian Ichter, and Sergey Levine. Navigation with large language models: Semantic guesswork as a heuristic for planning. In *Conference on Robot Learning*, pages 2683–2699. PMLR, 2023.

[55] Dhruv Shah and Sergey Levine. Viking: Vision-based kilometer-scale navigation with geographic hints. *arXiv preprint arXiv:2202.11271*, 2022.

[56] Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. Vint: A foundation model for visual navigation. *arXiv preprint arXiv:2306.14846*, 2023.

[57] Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. Mpnet: Masked and permuted pre-training for language understanding. *Advances in neural information processing systems*, 33:16857–16867, 2020.

[58] Ajay Sridhar, Dhruv Shah, Catherine Glossop, and Sergey Levine. Nomad: Goal masked diffusion policies for navigation and exploration. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 63–70. IEEE, 2024.

[59] Amr Suleiman, Zhengdong Zhang, Luca Carlone, Sertac Karaman, and Vivienne Sze. Navion: A 2-mw fully integrated real-time visual-inertial odometry accelerator for autonomous navigation of nano drones. *IEEE Journal of Solid-State Circuits*, 54(4):1106–1119, 2019.

[60] Leyuan Sun, Asako Kanezaki, Guillaume Caron, and Yusuke Yoshiyasu. Enhancing multimodal-input object goal navigation by leveraging large language models for inferring room–object relationship knowledge. *Advanced Engineering Informatics*, 65:103135, 2025.

[61] Andrew Szot, Alexander Clegg, Eric Undersander, Erik Wijmans, Yili Zhao, John Turner, Noah Maestre, Mustafa Mukadam, Devendra Singh Chaplot, Oleksandr Maksymets, et al. Habitat 2.0: Training home assistants to rearrange their habitat. *Advances in Neural Information Processing Systems*, 34:251–266, 2021.

[62] Emil Talpes, Douglas Williams, and Debjit Das Sarma. Dojo: The microarchitecture of tesla's exa-scale computer. In *2022 IEEE Hot Chips 34 Symposium (HCS)*, pages 1–28. IEEE Computer Society, 2022.

[63] Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. Gpt-4v (ision) for robotics: Multimodal task planning from human demonstration. *arXiv preprint arXiv:2311.12015*, 2023.

[64] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788, 2020.

[65] Xiaohan Wang, Yuhui Zhang, Orr Zohar, and Serena Yeung-Levy. Videoagent: Long-form video understanding with large language model as agent. *arXiv preprint arXiv:2403.10517*, 2024.

[66] Renjie Wei, Songqiang Xu, Linfeng Zhong, Zebin Yang, Qingyu Guo, Yuan Wang, Runsheng Wang, and Meng Li. Lightmamba: Efficient mamba acceleration on fpga with quantization and hardware co-design. In *2025 Design, Automation & Test in Europe Conference (DATE)*, pages 1–7. IEEE, 2025.

[67] Erik Wijmans, Abhishek Kadian, Ari Morcos, Stefan Lee, Irfan Essa, Devi Parikh, Manolis Savva, and Dhruv Batra. Dd-ppo: Learning near-perfect pointgoal navigators from 2.5 billion frames. *arXiv preprint arXiv:1911.00357*, 2019.

[68] Pengying Wu, Yao Mu, Bingxian Wu, Yi Hou, Ji Ma, Shanghang Zhang, and Chang Liu. Voronav: Voronoi-based zero-shot object navigation with large language model. *arXiv preprint arXiv:2401.02695*, 2024.

[69] Xiaodong Wu, Ran Duan, and Jianbing Ni. Unveiling security, privacy, and ethical concerns of chatgpt. *Journal of Information and Intelligence*, 2(2):102–115, 2024.

[70] Xiaoshi Wu, Feng Zhu, Rui Zhao, and Hongsheng Li. Cora: Adapting clip for open-vocabulary detection with region prompting and anchor pre-matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7031–7040, 2023.

[71] Guangxuan Xiao, Ji Lin, Mickael Seznec, Hao Wu, Julien Demouth, and Song Han. Smoothquant: Accurate and efficient post-training quantization for large language models. In *International Conference on Machine Learning*, pages 38087–38099. PMLR, 2023.

[72] Karmesh Yadav, Ram Ramrakhya, Arjun Majumdar, Vincent-Pierre Berges, Sachit Kuhar, Dhruv Batra, Alexei Baevski, and Oleksandr Maksymets. Offline visual representation learning for embodied navigation. In *Workshop on Reincarnating Reinforcement Learning at ICLR 2023*, 2023.

[73] Runming Yang, Taiqiang Wu, Jiahao Wang, Pengfei Hu, Yik-Chung Wu, Ngai Wong, and Yujiu Yang. Llm-neo: Parameter efficient knowledge distillation for large language models. *arXiv preprint arXiv:2411.06839*, 2024.

[74] Zebin Yang, Renze Chen, Taiqiang Wu, Ngai Wong, Yun Liang, Runsheng Wang, Ru Huang, and Meng Li. Mcubert: Memory-efficient bert inference on commodity microcontrollers. *arXiv preprint arXiv:2410.17957*, 2024.

[75] Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. Cacheblend: Fast large language model serving with cached knowledge fusion. *arXiv preprint arXiv:2405.16444*, 2024.

[76] Hang Yin, Xiuwei Xu, Zhenyu Wu, Jie Zhou, and Jiwen Lu. Sg-nav: Online 3d scene graph prompting for llm-based zero-shot object navigation. *Advances in Neural Information Processing Systems*, 37:5285–5307, 2024.

[77] Naoki Yokoyama and Sehoon Ha. Film-nav: Efficient and generalizable navigation via vlm fine-tuning. *arXiv preprint arXiv:2509.16445*, 2025.

[78] Naoki Yokoyama, Ram Ramrakhya, Abhishek Das, Dhruv Batra, and Sehoon Ha. Hm3d-ovon: A dataset and benchmark for open-vocabulary object goal navigation. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 5543–5550. IEEE, 2024.

[79] Bangguo Yu, Hamidreza Kasaei, and Ming Cao. L3mvn: Leveraging large language models for visual target navigation. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 3554–3560. IEEE, 2023.

[80] Wenxuan Zeng, Meng Li, Haichuan Yang, Wen-jie Lu, Runsheng Wang, and Ru Huang. Copriv: network/protocol co-optimization for communication-efficient private inference. *Advances in Neural Information Processing Systems*, 36:78906–78925, 2023.

[81] Beichen Zhang, Pan Zhang, Xiaoyi Dong, Yuhang Zang, and Jiaqi Wang. Long-clip: Unlocking the long-text capability of clip. In *European conference on computer vision*, pages 310–325. Springer, 2024.

[82] Jiazhao Zhang, Kunyu Wang, Shaoan Wang, Minghan Li, Haoran Liu, Songlin Wei, Zhongyuan Wang, Zhizheng Zhang, and He Wang. Uni-navid: A video-based vision-language-action model for unifying embodied navigation tasks. *arXiv preprint arXiv:2412.06224*, 2024.

[83] Jiazhao Zhang, Kunyu Wang, Rongtao Xu, Gengze Zhou, Yicong Hong, Xiaomeng Fang, Qi Wu, Zhizheng Zhang, and Wang He. Navid: Video-based vlm plans the next step for vision-and-language navigation. *arXiv preprint arXiv:2402.15852*, 2024.

[84] Jintao Zhang, Chendong Xiang, Haofeng Huang, Haocheng Xi, Jun Zhu, Jianfei Chen, et al. Spargeattention: Accurate and training-free sparse attention accelerating any model inference. In *Forty-second International Conference on Machine Learning*.

 https://doi.org/10.52202/085713-0152

[85] Junbo Zhang and Kaisheng Ma. Mg-vln: Benchmarking multi-goal and long-horizon vision-language navigation with language enhanced memory map. In *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7750–7757. IEEE, 2024.

[86] Sixian Zhang, Xinhang Song, Yubing Bai, Weijie Li, Yakui Chu, and Shuqiang Jiang. Hierarchical object-to-zone graph for object navigation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 15130–15140, 2021.

[87] Sixian Zhang, Xinhang Song, Xinyao Yu, Yubing Bai, Xinlong Guo, Weijie Li, and Shuqiang Jiang. Hoz++: Versatile hierarchical object-to-zone graph for object navigation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2025.

[88] Sixian Zhang, Xinyao Yu, Xinhang Song, Xiaohan Wang, and Shuqiang Jiang. Imagine before go: Self-supervised generative map for object goal navigation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16414–16425, 2024.

[89] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Jeff Huang, Chuyue Sun, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E Gonzalez, et al. Efficiently programming large language models using sglang. *arXiv preprint arXiv:2312.07104*, 2023.

[90] Shuzhang Zhong, Zebin Yang, Ruihao Gong, Runsheng Wang, Ru Huang, and Meng Li. Propd: Dynamic token tree pruning and generation for llm parallel decoding. In *Proceedings of the 43rd IEEE/ACM International Conference on Computer-Aided Design*, pages 1–8, 2024.

[91] Gengze Zhou, Yicong Hong, and Qi Wu. Navgpt: Explicit reasoning in vision-and-language navigation with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 7641–7649, 2024.

[92] Kaiwen Zhou, Kaizhi Zheng, Connor Pryor, Yilin Shen, Hongxia Jin, Lise Getoor, and Xin Eric Wang. Esc: Exploration with soft commonsense constraints for zero-shot object navigation. In *International Conference on Machine Learning*, pages 42829–42842. PMLR, 2023.

NeurIPS Paper Checklist

1. **Claims**

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

   Answer: [Yes]

   Justification: The main claims made in the abstract and introduction accurately reflect the paper's contributions and scope.

   Guidelines:

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

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitations of the work in Section 5 and Appendix I.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

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

   Answer: [NA]

https://doi.org/10.52202/085713-0152

Justification: This paper does not involve theoretical result.

Guidelines:

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

4. **Experimental result reproducibility**

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

   Answer: [Yes]

   Justification: We provide detailed information about experiments in the Section 4, appendix and code.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

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

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

Answer: [Yes]

Justification: We will release of code and data.

Guidelines:

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

6. **Experimental setting/details**

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

   Answer: [Yes]

   Justification: We specify all the training and test details in Section 4, appendix, and source code.

   Guidelines:

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

7. **Experiment statistical significance**

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

   Answer: [Yes]

   Justification: We repeated experiments with several runnings and found error bar is relatively small.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).

https://doi.org/10.52202/085713-0152

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

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

   Answer: [Yes]

   Justification: We provided sufficient information on the computer resources in the main text and appendix.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

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

   Answer: [Yes]

   Justification: The research conducted in the paper conformed, in every respect, with the NeurIPS Code of Ethics.

   Guidelines:

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

10. **Broader impacts**

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

    Answer: [Yes]

    Justification: This paper discusses both potential positive societal impacts and negative societal impacts of the work in the Appendix J.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

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

    Answer: [Yes]

    Justification: We describe safeguards for responsible release of models in the broader impacts section.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

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

    Answer: [Yes]

    Justification: We properly credited the creators or original owners of assets (e.g., code, data, models), used in the paper and conformed the license and terms.

    Guidelines:

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

https://doi.org/10.52202/085713-0152

- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

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

    Answer: [Yes]

    Justification: We communicated the details of the dataset/code/model as part of their submission.

    Guidelines:

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

14. **Crowdsourcing and research with human subjects**

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

    Answer: [NA]

    Justification: Our paper does not involve study participants.

    Guidelines:

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

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

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

    Answer: [NA]

    Justification: Our paper does not involve study participants.

    Guidelines:

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

- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [NA]

    Justification: Our paper does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.

https://doi.org/10.52202/085713-0152

A LLM inference and KV Cache

The generative inference process of LLMs is divided into two stages, the prefill stage and the decode stage. In the prefill stage, the prompt sequence is taken as input $\boldsymbol{X}$, and GEMMs are performed to compute the query ($\boldsymbol{Q} = \boldsymbol{X}\boldsymbol{W}_Q$), key ($\boldsymbol{K} = \boldsymbol{X}\boldsymbol{W}_K$), and value ($\boldsymbol{V} = \boldsymbol{X}\boldsymbol{W}_V$) matrices for each Transformer block. These computed key and value matrices are cached as KV cache for subsequent steps. The attention output is derived as $\mathrm{Attn}(\boldsymbol{Q}, \boldsymbol{K}, \boldsymbol{V}) = \mathrm{softmax}(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{D}})$, which further produces the first output token.

The decode stage iteratively generates one token per step via General Matrix-Vector operations. At step t, the newly generated token updates the cached Key and Value tensors through concatenation: $\boldsymbol{K}_t = \mathrm{concat}(\boldsymbol{K}_{t-1}, \boldsymbol{K}_{\mathrm{new}})$ and $\boldsymbol{V}_t = \mathrm{concat}(\boldsymbol{V}_{t-1}, \boldsymbol{V}_{\mathrm{new}})$. The updated cache enables efficient computation of self-attention over the extended sequence, yielding the next token. This autoregressive process continues until termination. Both stages leverage cached key-value pairs to avoid redundant computation.

B Object Navigation Flow

As discussed in Section 3.1, in our ObjNav, the goal-finding process is composed of iterative sub-goal finding tasks until the final goal is reached. Figure 11 shows an example of our ObjNAv. After reaching a sub-goal, the system generates semantic and distance information of the main objects from RGB-D sensor captures using detection models. Next, this semantic and spatial information is organized into the graph-based navigation map, and the updated map and goal instructions are provided to the LLM for goal planning. Here we also show an example of the LLM prompt and output in Section C.

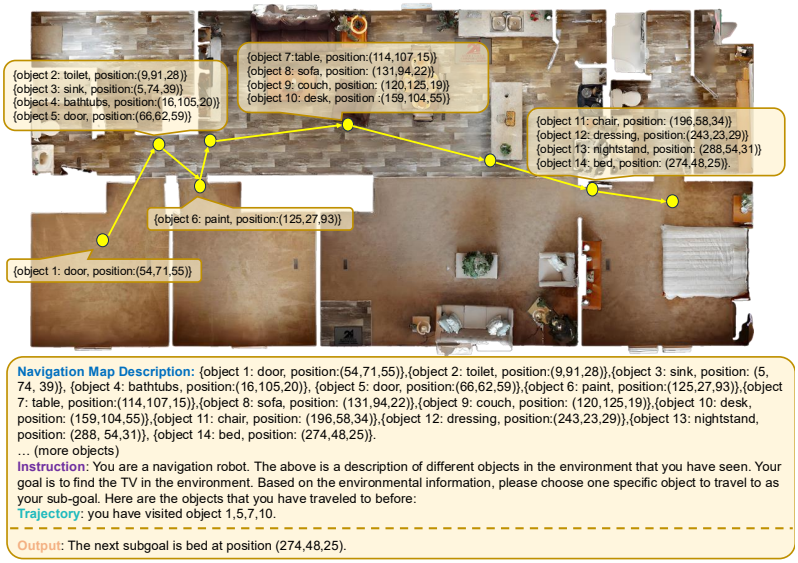


Figure 11: An example of our ObjNAv flow.

C Detailed Planning Example

We now further provide a more detailed example to illustrate the memory and prompt for EfficientNav.

1. Navigation Map Description: the navigation map records the objects and their position. To facilitate memory retrieval, we organize these objects into groups. Hence, an example of the navigation map is as follows:

https://doi.org/10.52202/085713-0152

Navigation Map Description

Object Group 1: {object: bathtubs, position:(54,71,55)}, {object: toilet, position:(9,91,28)}, {object: sink, position: (5, 74, 39)}, {object: towel, position:(16,105,20)} ...
Object Group 2: {object: door, position:(66,62,59)}, {object: paint, position:(125,27,93)}, {object: table, position:(114,107,15)}, {object: sofa, position:(131,94,22)}, {object: couch, position: (120,125,19)}, {object: desk, position: (159,104,55)}, {object: chair, position: (196,58,34)} ...
Object Group 3: {object: dressing, position:(243,23,29)}, {object: nightstand, position: (288, 54,31)}, {object: bed, position: (274,48,25)} ...
... (more groups)

Note that in order to reduce the computation cost of the navigation map, we compute the KV cache of each group only once and store the KV cache in memory. As the robot explores the environment, new objects will be added to different groups, whose KV cache will be updated accordingly.

2. Memory retrieval: for a given target, we then leverage semantic-aware memory retrieval to select relevant groups from the navigation map. Then, the KV cache of corresponding groups will be loaded to device memory. Thereby, memory retrieval not only reduces the memory loading cost but also helps the LLM planner to focus on the relevant environment to make better decisions. Let's assume the Group 2 and 3 in the navigation map above are selected.

3. Planning: Given the retrieved KV cache, we now query the LLM to plan for the exploration. The prompt for the LLM is as follows:

Prompt for Planning

Object Group 2: {object: door, position:(66,62,59)}, {object: paint, position:(125,27,93)}, {object: table, position:(114,107,15)}, {object: sofa, position: (131,94,22)}, {object: couch, position: (120,125,19)}, {object: desk, position: (159,104,55)}, {object: chair, position: (196,58,34)} ...
Object Group 3: {object: dressing, position:(243,23,29)}, {object: nightstand, position: (288, 54,31)}, {object: bed, position: (274,48,25)} ...
Instruction: You are a navigation robot. The above is a description of different objects in the environment that you have seen. Your final goal is to find the TV in the environment. Based on the environmental information, please choose one specific object to travel to as your sub-goal, following such format: "The next subgoal is xxx at position (xx, xx, xx)". Here are the objects that you have traveled to before:
Trajectory: You have visited the door at position (66,62,59) and the dressing at position (243,23,29).

Given the prompt above, the LLM generates the following instruction:

Planner Output

The next subgoal is sofa at position (131,94,22).

The robot will thereby follow the instruction to find the sofa at position (131, 94, 22). The process is repeated until the robot finds the target object.

D Network Architecture

We benchmark EfficientNav using four models: LLaVA-7B, LLaMA-3.2-11B, LLaVA-13B, and LLaVA-34B, each comprising an LLM backbone and a ViT-based vision encoder. As detailed in Table 7, their architectures scale systematically.

For LLM Backbones, LLaVA-7b, LLaMA-3.2-11b, LLaVA-13b, and LLaVA-34b adopt Mistral-7B-Instruct-v0.2, LLaMA-3.1, Vicuna-13B-v1.5, and Nous-Hermes-2-Yi-34B, respectively. Both the number of layers and hidden dimensions increase with model scale. For vision encoders, LLaVA

https://doi.org/10.52202/085713-0152

variants utilize a 24-layer ViT with a hidden dimension of 1024, while LLaMA-3.2-11B employs a deeper 32-layer ViT (hidden dimension of 1280). All vision encoders are fine-tuned during training.

Table 7: Network architecture

Model	LLM backbone			Vision Encoder	
	LLM	# Layers	Hidden Dim	# Layers	Hidden Dim
LLaVA-7b	Mistral-7B-Instruct-v0.2	32	4096	24	1024
LLaMA-3.2-11b	LLaMA-3.1	40	4096	32	1280
LLaVA-13b	Vicuna-13b-v1.5	40	5120	24	1024
LLaVA-34b	Nous-Hermes-2-Yi-34B	60	7168	24	1024

E CLIP Dependency

In semantics-aware retrieval, we use CLIP to remove redundant information and select relevant groups before sub-goal selection. CLIP is widely used in semantic matching [70, 81, 11]. In fact, as shown in Table 8, other semantic matching models can also be used and show reasonable performance (e.g.,MPNet[57], MiniLM [64]), showing the robustness of our method. While the LLM-based selection method shows high real-time latency because of the high computation cost, with no significant success rate improvement. This is because CLIP is enough to remove most of the redundant information and keep the most relevant groups. Even if a little redundant information is kept, the LLM planner can exclude it in the sub-goal selection period. So a light-weight semantic matching model is more reasonable.

Table 8: Ablation study on group selection method.

Method	SR	SPL	Real-time Latency
CLIP (ours)	80.0	41.5	0.87s
MPNet [57]	79.3	39.6	0.93s
MiniLM [64]	80.3	40.1	0.84s
LLM	80.7	42.1	6.55s

F Open-vocabulary Evaluation

In Section 4, we follow existing works use the HM3D dataset for comparison. However, we hope to emphasize that EfficientNav is not limited to HM3D and can be applied to open-vocabulary datasets. Here we evaluate our method on HM3D-OVON [78], which has more target object categories. The result is shown in Table 9, and our method also shows strong performance on open-vocabulary setting.

Table 9: Evaluation on HM3D-OVON.

Method	SR	SPL
Uni-NaVid [82]	41.3	21.1
FiLM-Nav [77]	44.9	24.5
EfficientNav (ours)	63.5	31.6

G Implementation Details

In EfficientNav, we use smaller open-source planners, such as LLaVA and LLaMA3.2. We use a single NVIDIA RTX A6000 GPU to deploy LLaVA-7b and LLaMA3.2-11b. When using LLaVA-13b and LLaVA-34b, we deploy our system on 2 and 4 NVIDIA RTX A6000 GPUs, respectively. We also deploy LLaVA-7b on a single Jetson AGX Orin. For the detection model, we use Grounding Dino [37].

H Comparison with Other Graph-based Method

Graph-based navigation maps are increasingly favored for their ability to explicitly represent the semantics and locations of visited objects. Among them, HOZ [86] proposes a hierarchical object-to-zone (HOZ) graph to provide coarse-to-fine guidance for navigation robots in unseen environments, which is constructed from scene, zone, and object nodes, updated online based on real-time observations. Based on HOZ, HOZ++ [87] introduces an adaptive graph structure and heterogeneous graph

fusion for the graph to further enhance object navigation performance. LROGNav [60] enhances navigation efficiency by integrating common-sense knowledge of object-to-room relationships extracted from large language models using both positive and negative chain-of-thought promptings, and by training a multi-channel Swin-UNet with multimodal inputs. Imagine Before Go [88] introduces a Self-supervised Generative Map (SGM) and improves exploration efficiency by learning explicit contextual relationships between objects and environments through self-supervised training. It leveraging both episodic observations and general knowledge from Large Language Models to generate unobserved regions of local semantic maps.

Our method EfficientNav improves the efficiency of LLM-based object goal navigation by navigation map caching and retrieval. Different from these works, our method focuses on the computation and management of KV cache for the navigation map description. We use discrete memory caching to decouple the context order and KV cache computation. For object grouping, we use attention-based memory clustering to reduce the impact of ignoring cross-attention. These methods enable KV cache reuse and significantly accelerate the prefilling stage of LLM planning.

I Limitations

In this section, we discuss the current limitations and potential avenues for future research.

First, in our method, we propose EfficientNav, enabling efficient zero-shot on-device ObjNAv. In fact, our memory caching and memory retrieval method also benefits the LLM inference on the cloud server by avoiding re-computation and removing redundant information. However, EfficientNav can not solve the problems of high communication latency or privacy concerns that cloud serving faces. Also, in our experiment, EfficientNav achieves 6.7$\times$ real-time latency reduction compared with GPT-4 planner. However, as the inference speed of LLM can not reach that of small models, EfficientNav should be carefully used in applications that need extremely low real-time latency.

J Broader Impacts

In our paper, our method enables efficient on-device zero-shot indoor ObjNav, overcoming the tight memory constraints by designing novel memory caching and retrieval mechanisms. However, our LLM-based ObjNav works in a zero-shot manner. This can avoid the heavy training cost, but the pre-trained LLM may not master some highly specialized knowledge. Therefore, in high-risk areas such as chemical laboratories with hazardous materials, EfficientNav should be used carefully.

https://doi.org/10.52202/085713-0152