HM3D-OVON: A Dataset and Benchmark for Open-Vocabulary Object Goal Navigation

Naoki Yokoyama[1*], Ram Ramrakhya[1*], Abhishek Das[2], Dhruv Batra[1], and Sehoon Ha[1]

***Abstract*— We present the Habitat-Matterport 3D Open Vocabulary Object Goal Navigation dataset (HM3D-OVON), a large-scale benchmark that broadens the scope and semantic range of prior Object Goal Navigation (ObjectNav) benchmarks. Leveraging the HM3DSem dataset, HM3D-OVON incorporates over 15k annotated instances of household objects across 379 distinct categories, derived from photo-realistic 3D scans of real-world environments. In contrast to earlier ObjectNav datasets, which limit goal objects to a predefined set of 6-21 categories, HM3D-OVON facilitates the training and evaluation of models with an open-set of goals defined through free-form language at test-time. Through this open-vocabulary formulation, HM3D-OVON encourages progress towards learning visuo-semantic navigation behaviors that are capable of searching for any object specified by text in an open-vocabulary manner. Additionally, we systematically evaluate and compare several different types of approaches on HM3D-OVON. We find that HM3D-OVON can be used to train an open-vocabulary ObjectNav agent that achieves both higher performance and is more robust to localization and actuation noise than the state-of-the-art ObjectNav approach. We hope that our benchmark and baseline results will drive interest in developing embodied agents that can navigate real-world spaces to find household objects specified through free-form language, taking a step towards more flexible and human-like semantic visual navigation. Code and videos available at: naoki.io/ovon.**

I. Introduction

Visual navigation to a language-specified object is an essential skill for robot assistants that can aid humans in a variety of tasks in indoor environments, such as "find my keys on the L-shaped couch". The interest in developing visual navigation systems has increased in recent years, highlighted by the embodied AI community's establishment of standardized evaluation metrics and benchmarks for numerous navigation tasks [1]–[3]. Various navigation tasks have been proposed, each defining goals differently – point-goal navigation for 2D coordinates [4], object-goal navigation [2], [5], image-goal navigation [6]–[8], and language-goal navigation (via referring expressions or step-by-step instructions) [9], [10]. In this work, we focus on the ObjectNav task where an agent is initialized in an indoor environment and tasked with navigating to an instance of a specified goal object category (*e.g.*, 'couch'). While existing ObjectNav benchmarks have typically concentrated on a limited, fixed set of object categories (6-21 object categories)

[1]NY, RR, DB, and SH are with the Georgia Institute of Technology; [2]AD is with Meta {nyokoyama, ram.ramrakhya, dbatra, sehoonha}@gatech.edu, abhshkdz@meta.com

[*]Denotes equal contribution.

Fig. 1: We study the Open-Vocabulary ObjectNav (OVON) task, which involves an agent tasked with navigating to object goals in an open-set, specified through language. In the above example, an agent is tasked with navigating to an 'L-Shaped Couch'.

and have only tested the generalization of navigation agents to novel environments, robotic agents in the real-world must also learn to generalize and navigate to an open set of object goal categories. To address this, we investigate the problem of open-vocabulary ObjectNav, where an agent will be asked to navigate to an object specified by language (seen or unseen during training). Fig. 1 illustrates an example of such an episode.

We introduce a dataset and benchmark named Habitat-Matterport 3D Open-Vocabulary ObjectNav (HM3D-OVON), designed to test the generalization of ObjectNav agents in an open-vocabulary setting using the HM3DSem [2] scene dataset. To examine how well agents can generalize to new goal object categories and environments, we propose three evaluation splits: 1) Val Seen- consists of goal categories seen during training, 2) Val Seen Synonyms- consists of goal categories synonymous to those seen during training (*i.e.*, "couch" category seen during training, evaluated on "sofa" during evaluation), 3) Val Unseen- consists of goal object categories that are not seen during training nor semantically similar to any category from the training set. We then use these evaluation splits to meticulously examine the performance of a variety of agents across goal object categories with varying degrees of semantic similarity to the training data.

We benchmark policies using several types of popular

learning paradigms on HM3D-OVON, including imitation learning (IL), reinforcement learning (RL), and modular methods [8], [11], [12], to understand their impact on the agent's ability to navigate to and recognize objects in an open-vocabulary setting. Through this benchmarking, we find that training end-to-end policies using imitation learning, specifically DAgger [13], with frontier exploration trajectories and fine-tuning with RL (referred to as DAgRL) outperforms all other end-to-end trained methods. However, DAgRL shows a drop in success rate of 11.9–23.0% compared to VAL SEEN when evaluating on VAL SEEN SYNONYMS and VAL UNSEEN, suggesting that trained methods struggle to generalize to unseen categories using end-to-end learning on ObjectNav alone. In contrast, the modular method VLFM [12], which leverages explicit maps and vision-language foundation models to explore the environment in a semantically meaningful way, achieves consistent performance between 32.4–35.2% on success rates across the three evaluation splits. We attribute this to the strong generalization capabilities of the open-vocabulary object detector (OWLv2 [14]) used to detect the goal object. Motivated by this observation, we find that augmenting DAgRL with an object detector and a navigation module for bee-lining to the detected object (referred to as DAgRL+OD) significantly improves the generalization of end-to-end methods on the VAL SEEN SYNONYMS and VAL UNSEEN splits, with a 9.6–18.8% increase in success rate. We also find that DAgRL is much more robust to noise that simulates real-world conditions than VLFM.

Additionally, we conduct a comprehensive analysis of different architectures used for encoding temporal information (transformer vs. RNN), imitation learning algorithms (behavioral cloning vs. DAgger), and types of trajectories used for imitation learning (frontier exploration vs. shortest path following). We find that policies perform significantly better when trained using a transformer instead of an RNN, with DAgger instead of behavioral cloning, and with frontier exploration instead of shortest path trajectories. Our findings on the impact of the types of trajectories used for imitation learning directly contradict the findings presented in SPOC [15], which asserted that shortest path trajectories lead to better performance than those that involve exploration for imitation learning. Furthermore, we present a detailed analysis of the failure modes of these agents, which illuminates the challenges and opportunities in developing robotic agents capable of robustly navigating to objects specified in free-form language in real-world environments, paving the way for more capable and generalizable visual semantic navigation robots. Code for HM3D-OVON can be found at naoki.io/ovon.

II. RELATED WORKS

ObjectNav in virtual environments. In recent years, several benchmarks have been established for training and evaluating a robot's ability to locate an instance of a given object category within a novel environment (ObjectNav). However, these benchmarks often exhibit two main shortcomings. First, the benchmarks may rely on a limited, fixed set of goal object categories for both training and evaluation. For instance, the HM3D ObjectNav dataset [20] encompasses only 6 different goal categories, while the MP3D ObjectNav dataset [17] includes 21. This restriction significantly hampers the training of approaches capable of navigating to a broader range of objects, and fails to test an approach's ability to generalize to new goal object categories unseen during training. Second, benchmarks may exclusively utilize synthetically generated scenes, rather than scans of real-world environments. For example, ProcTHOR [18] employs procedural generation to create floor plans and populate rooms with 3D assets, while OVMM [19] utilizes around 200 synthetic 3D scenes designed by humans. Although synthetic scene creation can produce a vast number of unique environments with less effort than scanning real-world scenes, the quality and realism can be significantly compromised; studies such as [21] have demonstrated that navigation agents trained on synthetic scenes exhibit poorer generalization to real-world-like environments (e.g., in terms of furniture quantity, types, and arrangement) compared to those trained on fewer, but meticulously designed synthetic scenes by human artists. To mitigate these issues, our work substantially expands the range of goal object categories, introduces different evaluation splits to assess how well an agent can generalize to new categories, and employs scans of furnished real-world scenes instead of synthetic ones. Table I contrasts our HM3D-OVON dataset with existing public benchmarks.

	Scene type	Object instances	Object categories
Habitat ObjectNav [16]	Real-world scans	7,599	6
MP3D ObjectNav [17]	Real-world scans	8,825	21
ProcTHOR [18]	Synthetic	1,633	108
OVMM [19]	Synthetic	7,892	150
HM3D-OVON	Real-world scans	15,661	379

TABLE I: Comparison of public ObjectNav benchmarks. Our HM3D-OVON benchmark provides a large number of unique objects and categories. HM3D-OVON also uses 3D scans of real-world environments instead of synthetic arrangements of 3D assets that better represent the semantic diversity of real-world conditions.

Methods for ObjectNav. Prior works on ObjectNav falls into two primary categories: modular approaches [11], [12], [22], and end-to-end learning via imitation or reinforcement learning [23]–[26]. Modular methods [11], [12], [22] break down the ObjectNav task into sub-skills such as exploration, recognition, and bee-lining (moving to the goal object once detected), employing specific components for each sub-skill. These approaches utilize heuristic-based exploration strategies, like frontier exploration [27], supported by explicit spatial and semantic maps, object detection and segmentation models for recognizing goal objects, and path planning algorithms like fast marching methods for waypoint navigation. End-to-end trained methods leverage neural networks to directly map sensor observations to actions.

Reinforcement learning (RL) variants of these methods learn exploration skills through hand-designed dense rewards [23], [24], while imitation learning (IL) methods draw on extensive human demonstrations [26], [28] to implicitly learn semantic exploration, or employ shortest path planners [15] within procedurally generated environments [18]. Prior studies employing end-to-end learning often use recurrent neural networks (RNNs) to encode temporal information as the agent navigates its environment [23]–[26], [28]. In contrast, our approach examines the use of transformers to encode observation history for end-to-end methods. Moreover, we conduct a thorough investigation into the impact of different imitation learning algorithms, types of trajectories used for imitation learning, and architectural choice to encode temporal information, offering an extensive comparison of these methods and guidelines for developing scalable and effective open-vocabulary ObjectNav agents.

III. The HM3D-OVON Benchmark

A. ObjectNav task definition

The ObjectNav task challenges an agent to locate any instance of a specified goal object category (*e.g.*, 'bed') within an unfamiliar environment [1]. At each time step, the agent receives a set of sensory inputs: an RGB image I_t, a depth image D_t, its relative displacement and heading from the start position (odometry) $P_t = (\Delta x, \Delta y, \Delta\theta)$, and the target object category G. The agent can select one of several actions: MOVE_FORWARD (by 0.25m), TURN_LEFT and TURN_RIGHT (by 30°), LOOK_UP and LOOK_DOWN (by 30°), and STOP actions. Success is defined as the agent invoking STOP within 1m of a goal object within 500 time steps. In our experiments, we configure the simulated agent to match the specifications of the Stretch robot [29], which has a height of 1.41m, a base radius of 17cm, and a 360×640 resolution RGB-D camera positioned at a height of 1.31m.

B. The HM3D-OVON dataset

We utilize the HM3DSem dataset's dense object annotations [30] to compile a vast collection of ObjectNav episodes, termed the HM3D-OVON dataset. HM3D-OVON includes 379 goal object categories across 181 unique, photorealistic virtual scans of real-world environments. We ensure goal objects are of significant size and visibility, occupying at least 5% of the Stretch's camera view from at least one vantage point within 1m of the object to affirm feasibility. The dataset is segmented into training and evaluation splits, with 145 scenes and 36 scenes, respectively, ensuring no scene or goal object instance overlap between splits. The training split features goal object instances across 280 categories, whereas the evaluation split comprises 178 categories.

To evaluate generalization to novel objects on varying levels, we divide the evaluation split into three smaller splits, each sharing the same scenes but utilizing mutually exclusive sets of goal object categories:

- VAL SEEN: uses goal object categories seen during training.
- VAL SEEN SYNONYMS: uses goal object categories semantically similar to those seen during training (*i.e.*, "couch" category seen during training, evaluated on "sofa" during evaluation).
- VAL UNSEEN: uses goal object categories semantically divergent from those encountered during training.

	Novel scenes?	Novel goal object categories?	Train similarity score range	Goal object categories
VAL SEEN	Yes	No	1.00	79
VAL SEEN SYNONYMS	Yes	Yes	[0.68, 0.96]	50
VAL UNSEEN	Yes	Yes	[0.45, 0.68]	49

TABLE II: All evaluation splits in HM3D-OVON use scenes and object instances unseen during training. VAL SEEN uses seen categories, VAL SEEN SYNONYMS uses unseen categories similar to a seen category, and VAL UNSEEN uses unseen categories not similar to any seen category. Similarity is determined with SentenceBERT.

For VAL SEEN SYNONYMS and VAL UNSEEN generation, we first uniformly sample ~25% of the object categories from HM3D-OVON. Then, we separate these categories using a semantic similarity metric calculated via SentenceBERT [31]. SentenceBERT, a fine-tuned variant of the pre-trained BERT network, is designed to gauge the semantic similarity between texts by comparing their embeddings' cosine similarity. We compute SentenceBERT embeddings for each sampled object category and evaluate its cosine similarity with all training split object categories. An object category is allocated to VAL SEEN SYNONYMS if it has a maximum similarity surpassing a threshold; otherwise, it is allocated to VAL UNSEEN. Table II summarizes the number of categories and the similarity ranges for each split.

C. Episode generation

An episode in HM3D-OVON comprises of a scene, the agent's starting position, and a goal object category. For episode creation, we first randomly select a goal object category and then randomly determine a starting position adhering to the following criteria: 1) at least one instance of the goal is on the same floor as the starting position, as stair climbing is not anticipated in indoor settings; and 2) the length of the shortest path to the nearest goal location must lie between 1m−30m. This approach aligns with the episode generation methodology of the ObjectNav task [20]. Fig. 1 illustrates a goal example for a single episode. Following this protocol, we generate $50k$ episodes per scene for the 145 training scenes, and $3k$ episodes per scene for the 36 validation scenes.

IV. HM3D-OVON Baselines

In this section, we compare various learning methodologies (imitation learning, reinforcement learning, and modular approaches) and architectural designs (transformer vs. RNN) as proposed in prior studies on object navigation. We evaluate each method on the HM3D-OVON benchmark.

Policy architecture. We employ frozen SigLIP [32] RGB and text encoders to encode the visual observations and the

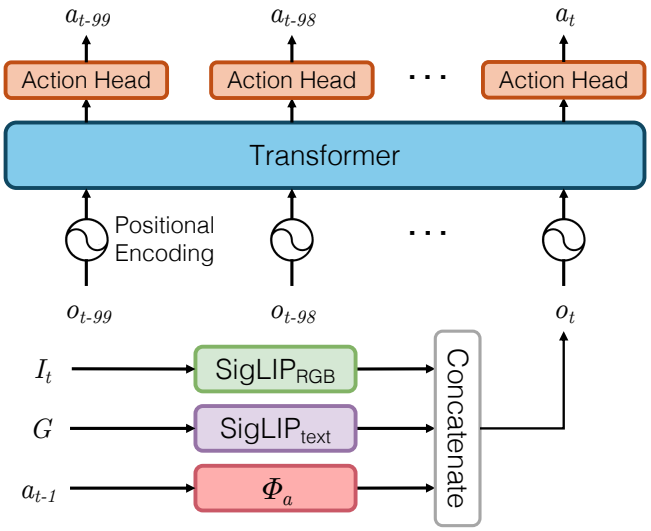

Fig. 2: Our OVON policy encodes the current visual observation I_t, the goal object category G, and the previous action a_{t-1} to form observation embedding o_t. At each step, the embedding sequence for the past 100 time steps is fed into a transformer, which uses an action head to sample an action a_t.

goal object category. These encoders have been identified as highly effective for ObjectNav by [15]. The encoders generate two 768-dimensional embeddings for the visual observation, $i_t = \text{SigLIP}_{\text{RGB}}(I_t)$, and the goal object category, $g = \text{SigLIP}_{\text{text}}(G)$. Additionally, the agent's previous action, a_{t-1}, is encoded into a 32-dimensional vector using an embedding layer, $p_t = \phi_a(a_{t-1})$. These embeddings are concatenated to form the observation embedding, $o_t = [i_t, g_t, p_t]$, which is fed into a 4-layer, decoder-only transformer [33] π_θ (8 heads, hidden size of 512), with a maximum context length of 100. π_θ takes in the past 100 consecutive observations $[o_{t-99}, ..., o_t]$ and outputs a feature vector for the current time step. This vector is passed through a linear layer (action head) that predicts a categorical distribution from which an action a_t is sampled, $a_t \sim \pi_\theta(\cdot|o_{t-99}, ..., o_t)$. During RL, an additional linear layer (critic head) is used to project the feature vector into a value estimate for the current state.

When comparing against RNN-based policies, the only architectural change we make is replacing the transformer with a 4-layer LSTM [34] with a similar parameter count, for fair comparison.

Behavioral cloning (BC). Learning from demonstrations has been shown to be a powerful approach for developing efficient semantic navigation behaviors [15], [26], [28], [35]. Behavioral cloning employs supervised learning on a dataset of observation-action pairs from expert demonstrations to train policies. Consider a policy π_θ parameterized by θ that maps observations o_t to an action distribution, $\pi_\theta(\cdot|o_t)$. Let τ denote a demonstration consisting of observation-action pairs, $\tau = [(o_0, a_0), (o_1, a_1), ..., (o_n, a_n)]$, and $T = \{\tau_i\}_i^n$ denote a dataset of demonstrations. The objective function optimization can be described as:

$$\theta^* = \arg\max_\theta \sum_{i=1}^{N} \sum_{(o_t, a_t) \in \tau_i} \log\Big(\pi_\theta(a_t \mid o_t)\Big)$$

While prior work showed that behavioral cloning using demonstrations collected from humans performing the ObjectNav task can train effective policies [26], [28], these demonstrations were limited in diversity and are expensive to collect, especially for the amount of categories in HM3D-OVON. SPOC [15] used a path planner to generate the shortest possible obstacle-free trajectory from the start pose to the goal object, and demonstrated that these trajectories lead to better results than those generated by an expert that exhibits more exploration. However, this directly contradicts the results of [28], which show learning from demonstrations that involve frontier exploration yield better performing policies than shortest path trajectories. These trajectories are generated by executing frontier-based exploration, which involves the agent systematically moving towards unexplored areas ('frontiers') of the environment, until a goal object is within range (3.5m in our experiments), at which point a shortest path planner is used to plan a path to the goal object ('bee-line'). In this work, we experiment with learning from either frontier exploration or shortest path trajectories. For behavioral cloning, we generate a trajectory for each episode, for each of the two types (7.25 million trajectories for each type).

DAgger. DAgger [13] is a supervised learning algorithm that adopts the same loss function as behavioral cloning. However, unlike behavioral cloning, DAgger involves an expert who provides new action labels for the agent's trajectories 'online' during training. Additionally, the action generated by the policy π_θ is utilized to advance the environment, rather than the expert's actions. The formulation of DAgger's learning algorithm is similar to behavioral cloning, except each labeled trajectory $\hat{\tau}$ now consists of observation-action pairs $\hat{\tau} = [(o_0^\pi, \hat{a}_0), (o_1^\pi, \hat{a}_1), ..., (o_n^\pi, \hat{a}_n)]$, where action labels $\hat{a}_t$ are provided by the expert, given o_t^π.

Unlike behavioral cloning, which relies on a pre-recorded dataset, DAgger generates o_t^π (and consequently, $\hat{a}_t$) using a_{t-1}^π while π_θ is updated, necessitating the capability to interact with the environment and consult the expert online during learning. Thus, it is crucial to use an expert that can swiftly provide a label $\hat{a}_t$ for o_t^π, as a slow expert can substantially reduce the speed of training. Existing implementations of frontier-based trajectory generation for ObjectNav [11], [28] are overly slow for in-the-loop execution, taking 250ms per time step, leading to a separation of trajectory generation and supervised learning into discrete stages to maintain training speed. This constraint has forced prior studies like [28] to adopt behavioral cloning for learning from offline-collected frontier exploration trajectories. To counteract this, we introduce, alongside our benchmark, an implementation for frontier-based exploration and bee-lining engineered specifically for rapid execution at each time step, ensuring minimal impact on training speed. Our method only requires

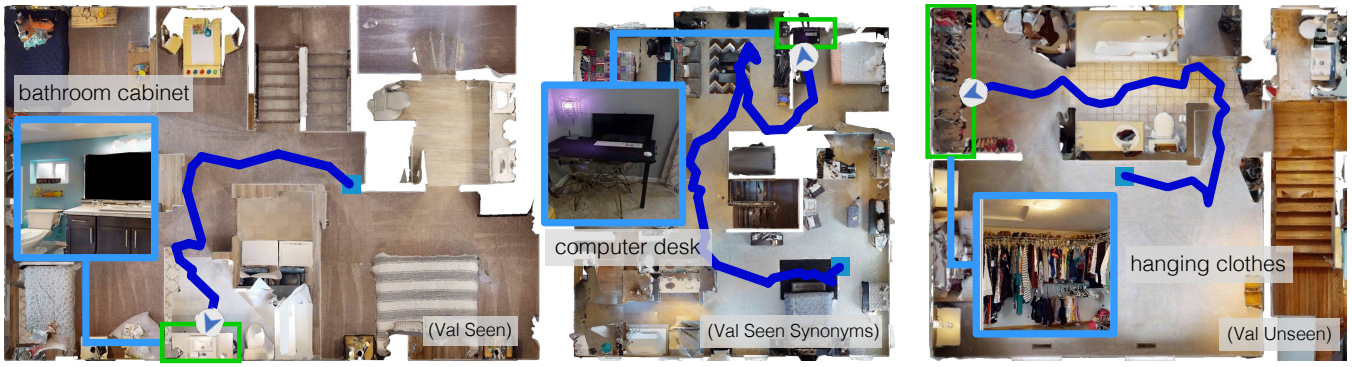

Fig. 3: Examples of successes of our DAgRL policy for each evaluation split. DAgRL can efficiently explore the environment, avoid obstacles, and stop in front of the goal object when it is spotted, using only RGB observations. Videos can be found at naoki.io/ovon.

an average of 1ms per time step to suggest the appropriate action for guiding the agent from its present position toward the next frontier (or the goal object, if close enough). This not only facilitates real-time generation of ground-truth expert demonstrations, but also frees the agent from adhering to predetermined trajectories, eliminating the need for a distinct trajectory generation phase prior to training.

Reinforcement learning (RL). To train a policy using RL, we use PPO [36] with a shaped navigation reward commonly used for the PointGoal Navigation task [37]. This reward consists of Δ_{dtg} which is the change in the agent's geodesic distance to the nearest goal between time step t and $t-1$, and $r_{success}$ (set to 2.5), the reward received upon success:

$$r_t = \begin{cases} r_{success} & \text{if success,} \\ -\Delta_{dtg} - 0.01 & \text{otherwise} \end{cases}$$

BCRL. We pretrain OVON policies using behavior cloning and fine-tune using reinforcement learning with sparse rewards, following prior work [28].

DAgRL. Similar to BCRL, we pretrain OVON policies using DAgger and fine-tune using reinforcement learning with sparse rewards.

VLFM. Vision-Language Frontier Maps (VLFM) [12] is a modular method that achieves state-of-the-art performance on various ObjectNav benchmarks, and can also support open-vocabulary goal object categories. VLFM leverages the depth and odometry sensors to build an occupancy map as the robot explores the environment. In parallel, VLFM utilizes a vision-language foundation model to assess the semantic significance of each explored area in relation to the target object category. While prioritizing semantically significant areas, VLFM exhaustively explores the environment until an object detector has detected a goal object. Once a goal object has been detected, the depth camera is used to estimate its coordinates. The agent then utilizes a policy trained specifically for reaching a given goal coordinate using D_t and P_t to reach the object and call STOP. We modify the original VLFM implementation by integrating OWLv2 [14] as the open-vocabulary object detector, as we found it to outperform GroundingDINO [38] in our experiments.

DAgRL+OD. This approach extends the DAgRL baseline by incorporating the OWLv2 object detector. It relies on a trained DAgRL policy to explore the environment until the detector detects a goal object. Upon detection, the agent transitions from the DAgRL policy to a trained point-goal policy (same as the one used in VLFM) for bee-lining to the detected object.

Training details. The BC and DAgger policies were trained for 150M steps, as incremental improvements diminished beyond this point. BCRL and DAgRL received additional fine-tuning for 150M steps using PPO. To make comparison with these two fine-tuned policies fair, the RL policy is trained for 300M steps. All policies were trained across 16 environments per GPU, distributed over 8 TITAN Xp GPUs utilizing Variable Experience Rollout [39].

V. Results

In this section, we aim to address the following questions:

1) What differences in performance can be observed between policies trained using different learning methods (RL, IL, or modular learning)?
2) To what extent do various trajectory generation strategies influence the success of imitation learning policies?
3) How do different state encoder architectures (transformers vs. RNNs) impact performance?
4) How robust are our baselines to noise that simulate real-world conditions?

A. Benchmarking learning methods

We compare different learning-based approaches: end-to-end policies trained using RL, BC, DAg, BCRL, and DAgRL, as well as a modular approach, VLFM [12], in Table III. We report two metrics – success rate (SR) and Success weighted by Path Length (SPL) [40], to compare the performance of baselines, across three seeds. We observe that the policy pretrained using DAgger with frontier exploration trajectories and fine-tuned using RL with sparse rewards, *i.e.*, DAgRL (row 5), outperforms all other end-to-end baselines that do not use a navigation module for bee-lining (rows 1-4). We attribute this performance to the effective exploration behavior learned using imitation learning from frontier exploration trajectories, and RL fine-tuning that improves the policy's ability to select frontiers more likely to lead to the goal object. Surprisingly, we find policies trained using RL

Method	Val Seen		Val Seen Synonyms		Val Unseen	
	SR(↑)	SPL(↑)	SR(↑)	SPL(↑)	SR(↑)	SPL(↑)
1) BC	11.1±0.1	4.5±0.1	9.9±0.4	3.8±0.1	5.4±0.1	1.9±0.2
2) DAgger	18.1±0.4	9.4±0.3	15.0±0.4	7.4±0.3	10.2±0.5	4.7±0.3
3) RL	39.2±0.4	18.7±0.2	27.8±0.1	11.7±0.2	18.6±0.3	7.5±0.2
4) BCRL	20.2±0.6	8.2±0.4	15.2±0.1	5.3±0.1	8.0±0.2	2.8±0.1
5) DAgRL	**41.3**±0.3	**21.2**±0.3	29.4±0.3	14.4±0.1	18.3±0.3	7.9±0.1
6) *VLFM [12]	35.2	18.6	32.4	17.3	35.2	19.6
7) DAgRL+OD	38.5±0.4	21.1±0.4	**39.0**±0.7	**21.4**±0.5	**37.1**±0.2	**19.9**±0.3

*deterministic method

TABLE III: Performance of all baselines on the three evaluation splits of HM3D-OVON. DAgRL+OD outperforms the state-of-the-art ObjectNav approach, VLFM.

with a distance to goal reward (row 3) using a transformer state encoder performs comparably to the DAgRL baseline (row 5), only up to 2.1% worse on SR and up to 2.7% worse on SPL. Next, we compare the performance of end-to-end trained methods with a modular, state-of-the-art approach, VLFM [12] (row 6). While VLFM performs worse than DAgRL on the Val Seen split by 6.1% on SR and 2.6% on SPL, it outperforms DAgRL on the Val Seen Synonyms and Val Unseen splits by 3.0−16.9% on SR and 2.9−11.7% on SPL, demonstrating strong generalization to unseen object categories. We find this gap in performance is due to VLFM's use of an open-vocabulary object detector for identifying the goal object. This detector was trained on a substantially larger set of object classes, which may overlap with classes present in the evaluation splits of HM3D-OVON. To remedy this issue, DAgRL+OD (row 7) is equipped with an open-vocabulary object detector and the same point-goal policy as VLFM for bee-lining to detected goal objects. We find that DAgRL+OD outperforms VLFM on all three evaluation splits by 1.9−6.6% on SR and 0.3−4.1% on SPL. A characterization of this issue and other failures modes are analyzed in Sec.V-E.

B. Comparison of trajectories used for imitation learning

We compare different types of trajectories (frontier exploration vs. shortest path) used during imitation learning. We find that policies trained using frontier exploration generally outperform those trained using shortest path following, significantly. Our results are summarized in Table IV; in rows 1, 2, and 4, using frontier exploration rather than shortest path following improves SR and SPL by 4.6−6.0% and 1.0−2.2%, respectively. For row 3, BCRL, we found that the poor performance of the pretrained BC policies led to suboptimal sparse RL fine-tuning, resulting in performance that fails to surpass ~20% in SR for both frontier exploration and shortest path trajectory training.

The observation that trajectories that incorporate exploration are more effective for imitation learning than shortest path trajectories directly contradicts the findings presented by SPOC [15]. We attribute this to two possible factors: first, the exploration trajectories in [15] are different from our frontier exploration trajectories. While we designate an area of the environment as explored once it has appeared in the agent's field-of-view within a certain distance (3m), the exploration trajectories used in [15] iteratively visits the next closest room and navigates towards every object within it until at least 75% of the objects in the room have been seen. This could consume more steps compared to the more direct method of filling out the map used in frontier exploration, resulting in less efficient learning. Additionally, its emphasis on objects within a room might not generalize well across different environments, especially if the environments vary widely in layout, room size, or object density. This contrasts with frontier exploration, which is more focused on spatial layout and could generalize better. The second possible factor is the difference in the navigation complexity of the environments used. [15] used synthetic ProcThor [18] scenes, which have been observed to contain much less obstacles and smaller floor plans than real-world scans from HM3D [21]. The higher complexity of HM3D scenes may make policies that learn from trajectories that exhibit exploration behaviors perform better than policies that only learn from shortest paths.

#	Method	Shortest Path		Frontier Exploration	
		SR (↑)	SPL (↑)	SR (↑)	SPL (↑)
1)	BC	5.1±0.1	3.5±0.1	**11.1**±0.1	**4.5**±0.1
2)	DAgger	13.4±0.2	8.0±0.1	**18.1**±0.4	**9.4**±0.3
3)	BCRL	**20.6**±0.3	**8.5**±0.2	20.2±0.6	8.2±0.4
4)	DAgRL	37.7±0.6	19.0±0.4	**41.3**±0.3	**21.2**±0.3

TABLE IV: Val Seen performance comparing trajectories used for imitation learning. We find policies trained using frontier-exploration perform better than those trained using shortest paths.

C. Transformer vs. RNN

We examine various techniques for incorporating temporal information from preceding time steps to train effective OVON policies. The findings, as presented in Table V, demonstrate that transformer-based policies tend to surpass RNN-based counterparts (rows 2, 3, and 5), achieving improvements of up to 6.3% and 4.5% in SR and SPL, respectively. The exceptions, which do not exhibit significant performance enhancement, are the baselines employing behavioral cloning rather than DAgger (rows 1 and 4). This discrepancy is likely due to the comparative lack of training data diversity in BC when contrasted with DAgger. DAgger and RL enable the agent to explore previously unencountered trajectories as the policy evolves during training, whereas BC restricts the policy to learning solely from teacher policy trajectories, which remain static throughout the training process. Given that transformer performance scales more favorably with training data volume, the limited diversity in the training data for behavioral cloning might explain why transformer-based policies did not significantly outperform those based on RNNs.

D. Robustness to noise

In perceiving the environment, the policies we train for ObjectNav rely solely on an RGB sensor. This makes them

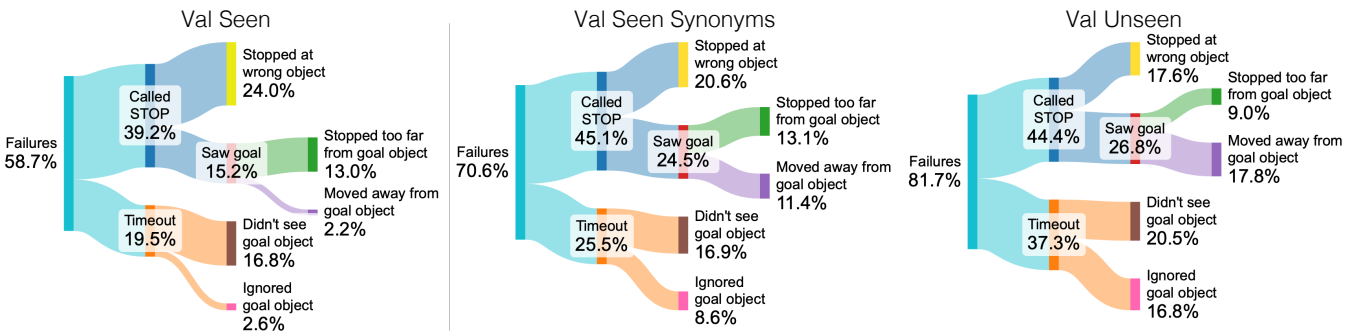

Fig. 4: Failure analysis of DAgRL. As the goal object categories become less similar to those seen in training (*i.e.*, VAL SEEN, VAL UNSEEN), the agent more frequently fails from timeouts (never calling STOP) and more frequently ignores the goal object.

		RNN		TRANSFORMER	
#	Method	SR (↑)	SPL (↑)	SR (↑)	SPL (↑)
1)	BC	10.5±0.3	4.1±0.1	**11.1±0.1**	**4.5±0.1**
2)	DAgger	14.7±0.0	7.4±0.2	**18.1±0.4**	**9.4±0.3**
3)	RL	32.9±0.4	15.5±0.2	**39.2±0.4**	**18.7±0.2**
4)	BCRL	**21.0±0.4**	**11.1±0.2**	20.2±0.6	8.2±0.4
5)	DAgRL	36.5±0.8	16.7±0.3	**41.3±0.3**	**21.2±0.3**

TABLE V: Performance on VAL SEEN comparing transformer- and RNN-based architectures for processing temporal information. Transformer-based policy architectures generally perform the best.

Method	Eval Noise	VAL SEEN		VAL SEEN SYNONYMS		VAL UNSEEN	
		SR(↑)	SPL(↑)	SR(↑)	SPL(↑)	SR(↑)	SPL(↑)
VLFM*	-	35.2	18.6	32.4	17.3	35.2	19.6
	✓	28.8±0.2	14.0±0.1	28.5±0.2	13.9±0.1	29.1±0.3	14.2±0.1
DAgRL+OD	-	38.5±0.4	21.1±0.4	39.0±0.7	21.4±0.5	37.1±0.2	19.9±0.3
	✓	38.2±1.0	18.1±0.4	38.9±0.5	18.8±0.4	36.9±0.6	17.4±0.3

*deterministic method when evaluated with no noise

TABLE VI: Despite being trained without noise, our DAgRL+OD policy is robust against odometry and actuation noise introduced at test-time, since it only relies on visual sensors.

inherently less sensitive to types of perturbations that may affect other approaches that rely on building maps upon which observations are spatially projected using odometry and depth sensors. We evaluate our best baseline, DAgRL+OD, and VLFM in the presence of noise that simulates real-world conditions. We use an actuation noise model within Habitat acquired from mocap-based benchmarking recorded using a robot with a dynamics model and sensor suite similar to the Stretch [41]. This adds Gaussian noise to the position and rotation of the robot each time it takes an action. We also add noise uniformly sampled sensor noise using values drawn from the uncertainty study done by [42], adding between $\pm$7mm and $\pm 2^\circ$ to the odometry sensor P_t at each time step. Note that neither DAgRL+OD nor VLFM use models that were trained in the presence of these types of noise.

The results of evaluation with noise are shown in Table VI. We find that VLFM is sensitive to noise, and drops in SR and SPL by up to 6.1% and 5.4%, respectively. This can be attributed to the fact that VLFM iteratively builds top-down maps that it relies on to decide where to explore next, and the added noise can make the maps innaccurate. In contrast, DAgRL+OD is robust to noise, and drops in SR and SPL by only up to 0.3% and 3.0% respectively. Since the DAgRL policy used for exploration is only able to use RGB observations from a sequence of previous consecutive time steps to determine where it currently is relative to where it has been before, the policy does *not* learn to rely on precise localization to attain good performance. Thus, it is robust to the additional noise that is injected at test-time. The larger drop in SPL compared to SR can be attributed to the fact that the noise causes the robot to deviate from its desired path, and the additional corrective actions required leads to longer paths compared to evaluation without noise.

E. Failure analysis

To characterize the types of behavior that can be learned from HM3D-OVON, we present a detailed analysis of the different failure modes of DAgRL, the policy that attains the best performance without relying on an external open-vocabulary object detector. A breakdown of the different failures modes on each of the three evaluation splits is visualized in Fig. 4. As the goal object categories in the evaluation split decrease in similarity to those seen during training (VAL SEEN →VAL SEEN SYNONYMS →VAL UNSEEN), the following trends occur: We observe that failures that occur from timeout (*i.e.*, never calling STOP) increase (orange in Fig. 4, 19.5%→25.5%→37.3%), primarily due to ignoring the goal object (pink, 2.6%→8.6%→16.8%), rather than inefficient exploration that causes the agent to miss the goal object (brown, stays between 16.8%-20.5%). We also observe that the agent is more likely to move away from the goal object (enter and leave its success region) and call stop elsewhere (purple, 2.2%→11.4%→17.8%). These trends indicate that the agent struggles to generalize and correctly navigate to and call STOP when seeing a goal object that is too semantically different from the goal categories in HM3D-OVON's training split. However, as shown by the performance of DAgRL+OD, these shortcomings can be addressed using a pretrained open-vocabulary object detector and a way to navigate the robot to a detected object and call STOP (*i.e.*, a point-goal policy).

The causes of failure that consistently contribute a large portion of the failure across all three of the evaluation splits are: (1) stopping at the wrong object (yellow, between

16.8%-20.5%), and (2) not seeing the goal object (brown, between 16.8%-20.5%). This indicates that future work should prioritize reducing false positives and improving exploration to find the goal object within the time step limit.

VI. Conclusion

We present HM3D-OVON, a large-scale dataset and benchmark that provides 379 goal object categories and over $15k$ annotated instances of household objects across 181 unique, photo-realistic virtual scans of real-world environments. HM3D-OVON facilitates the training and evaluation of models with an open-set of goals defined through free-form language, compared to previous datasets that are limited to a predefined set of object categories at test-time. Through extensive experiments, we demonstrate that HM3D-OVON can be used to train an open vocabulary ObjectNav agent that achieves both higher performance and better robustness to localization and actuation noise than the state-of-the-art. We hope that HM3D-OVON leads to further advancements in embodied AI and opens up new avenues for research in visual semantic navigation and object recognition.

VII. Acknowledgements

The Georgia Tech effort was supported in part by Hyundai Mobis, PONR YIPs, and ARO PECASE. The views and conclusions are those of the authors and should not be interpreted as representing the U.S. Government, or any sponsor.

References

[1] D. Batra, A. Gokaslan, A. Kembhavi, O. Maksymets, R. Mottaghi, *et al.*, "ObjectNav revisited: On evaluation of embodied agents navigating to objects," *arXiv preprint arXiv:2006.13171*, 2020.

[2] K. Yadav, R. Ramrakhya, S. K. Ramakrishnan, T. Gervet, J. Turner, *et al.*, "Habitat-matterport 3d semantics dataset," in *CVPR*, June 2023.

[3] S. Wani, S. Patel, U. Jain, A. X. Chang, and M. Savva, "Multion: Benchmarking semantic map memory using multi-object navigation," 2020.

[4] M. Savva, A. Kadian, O. Maksymets, Y. Zhao, E. Wijmans, *et al.*, "Habitat: A platform for embodied ai research," in *ICCV*, 2019.

[5] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, *et al.*, "Robothor: An open simulation-to-real embodied ai platform," in *CVPR*, 2020.

[6] Y. Zhu, R. Mottaghi, E. Kolve, J. J. Lim, A. Gupta, *et al.*, "Target-driven visual navigation in indoor scenes using deep reinforcement learning," in *ICRA*, 2017.

[7] J. Krantz, T. Gervet, K. Yadav, A. Wang, C. Paxton, *et al.*, "Navigating to objects specified by images," in *ICCV*, 2023.

[8] D. S. Chaplot, R. Salakhutdinov, A. Gupta, and S. Gupta, "Neural topological slam for visual navigation," in *CVPR*, 2020.

[9] Y. Qi, Q. Wu, P. Anderson, X. Wang, W. Y. Wang, *et al.*, "Reverie: Remote embodied visual referring expression in real indoor environments," in *CVPR*, 2020.

[10] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, *et al.*, "Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments," in *CVPR*, 2018.

[11] D. S. Chaplot, D. Gandhi, A. Gupta, and R. Salakhutdinov, "Object goal navigation using goal-oriented semantic exploration," in *In Neural Information Processing Systems*, 2020.

[12] N. Yokoyama, S. Ha, D. Batra, J. Wang, and B. Bucher, "Vlfm: Vision-language frontier maps for zero-shot semantic navigation," in *International Conference on Robotics and Automation (ICRA)*, 2024.

[13] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011, pp. 627–635.

[14] M. Minderer, A. Gritsenko, and N. Houlsby, "Scaling open-vocabulary object detection," in *NeurIPS*, 2024.

[15] E. Kiana, G. Tanmay, H. Rose, S. Jordi, W. Luca, *et al.*, "Imitating shortest paths in simulation enables effective navigation and manipulation in the real world," *arXiv*, 2023.

[16] K. Yadav, J. Krantz, R. Ramrakhya, S. K. Ramakrishnan, J. Yang, *et al.*, "Habitat challenge 2023," https://aihabitat.org/challenge/2023/, 2023.

[17] A. Chang, A. Dai, T. Funkhouser, M. Halber, M. Niessner, *et al.*, "Matterport3D: Learning from RGB-D Data in Indoor Environments," *International Conference on 3D Vision (3DV)*, 2017.

[18] D. Matt, V. Eli, H. Alvaro, W. Luca, S. Jordi, *et al.*, "Procthor: Large-scale embodied ai using procedural generation," in *NeurIPS*, 2022.

[19] S. Yenamandra, A. Ramachandran, K. Yadav, A. Wang, M. Khanna, *et al.*, "Homerobot: Open vocab mobile manipulation," 2023.

[20] H. Team, "Habitat challenge, 2022," https://aihabitat.org/challenge/2022, 2020.

[21] M. Khanna*, Y. Mao*, H. Jiang, S. Haresh, B. Shacklett, *et al.*, "Habitat Synthetic Scenes Dataset (HSSD-200): An Analysis of 3D Scene Scale and Realism Tradeoffs for ObjectGoal Navigation," *arXiv preprint*, 2023.

[22] S. K. Ramakrishnan, D. S. Chaplot, Z. Al-Halah, J. Malik, and K. Grauman, "PONI: Potential Functions for ObjectGoal Navigation with Interaction-free Learning," in *CVPR*, 2022.

[23] J. Ye, D. Batra, A. Das, and E. Wijmans, "Auxiliary tasks and exploration enable objectnav," in *ICCV*, 2021.

[24] O. Maksymets, V. Cartillier, A. Gokaslan, E. Wijmans, W. Galuba, *et al.*, "THDA: Treasure Hunt Data Augmentation for Semantic Navigation," in *ICCV*, 2021.

[25] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi, "Simple but effective: Clip embeddings for embodied ai," in *CVPR*, June 2022.

[26] R. Ramrakhya, E. Undersander, D. Batra, and A. Das, "Habitat-web: Learning embodied object-search strategies from human demonstrations at scale," in *CVPR*, 2022.

[27] B. Yamauchi, "A Frontier-Based Approach for Autonomous Exploration," in *Proc. IEEE Int. Symp. Comput. Intell. Robot. Autom., CIRA*. IEEE, 1997, pp. 146–151.

[28] R. Ramrakhya, D. Batra, E. Wijmans, and A. Das, "Pirlnav: Pretraining with imitation and rl finetuning for objectnav," in *CVPR*, 2023.

[29] C. C. Kemp, A. Edsinger, H. M. Clever, and B. Matulevich, "The design of stretch: A compact, lightweight mobile manipulator for indoor human environments," in *ICRA*. IEEE, 2022, pp. 3150–3157.

[30] K. Yadav, R. Ramrakhya, S. K. Ramakrishnan, T. Gervet, J. Turner, *et al.*, "Habitat-matterport 3d semantics dataset," in *CVPR*, 2023.

[31] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proc. EMNLP*. ACL, 2019.

[32] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," *arXiv*, 2023.

[33] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, *et al.*, "Attention is all you need," in *NIPS*, 2017.

[34] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[35] K. Yadav, A. Majumdar, R. Ramrakhya, N. Yokoyama, A. Baevski, *et al.*, "Ovrl-v2: A simple state-of-art baseline for imagenav and objectnav," *arXiv preprint arXiv:2303.07798*, 2023.

[36] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv:1707.06347*, 2017.

[37] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, *et al.*, "DD-PPO: Learning near-perfect pointgoal navigators from 2.5 billion frames," in *ICLR*, 2020.

[38] S. Liu, Z. Zeng, T. Ren, F. Li, H. Zhang, *et al.*, "Grounding dino: Marrying dino with grounded pre-training for open-set object detection," *arXiv preprint arXiv:2303.05499*, 2023.

[39] E. Wijmans, I. Essa, and D. Batra, "Ver: Scaling on-policy rl leads to the emergence of navigation in embodied rearrangement," *Advances in Neural Information Processing Systems*, vol. 35, pp. 7727–7740, 2022.

[40] P. Anderson, A. X. Chang, D. S. Chaplot, A. Dosovitskiy, S. Gupta, *et al.*, "On evaluation of embodied navigation agents," *arXiv preprint arXiv:1807.06757*, 2018.

[41] A. Murali, T. Chen, K. V. Alwala, D. Gandhi, L. Pinto, *et al.*, "Pyrobot: An open-source robotics framework for research and benchmarking," *arXiv preprint arXiv:1906.08236*, 2019.

[42] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, *et al.*, "Sim2Real predictivity: Does Evaluation in Simulation Predict Real-world Performance?" in *IEEE Robotics and Automation Letters (RA-L)*, 2020.