ZSON: Zero-Shot Object-Goal Navigation using Multimodal Goal Embeddings

Anonymous Author(s)
Affiliation
Address
email

Abstract

We present a scalable approach for learning *open-world* object-goal navigation (ObjectNav). Our approach is *zero-shot* – i.e., it does not require ObjectNav annotations for training. Instead, we convert the image-goal navigation task into semantic-goal navigation (SemanticNav) by encoding goal images into a multi-modal, semantic embedding space. SemanticNav agents can be trained at scale in unannotated 3D environments (e.g., HM3D). After training, the resulting agents can navigate to objects described in free-form natural language (e.g., *"sink," "bathroom sink,"* etc.), reflecting the requirements for open-world ObjectNav. We extensively evaluate our agents on three ObjectNav datasets (Gibson, HM3D, and MP3D) and observe absolute improvements in success of 4.2% - 20.0% over existing zero-shot methods. In an open-world setting, we discover that our agents can generalize to compound instructions with a room explicitly mentioned (e.g., *"Find a kitchen sink"*) and when the target room can be inferred (e.g., *"Find a sink and a stove"*).

1 Introduction

Imagine asking a home assistant robot to fetch a *"flat-head screwdriver"* or the *"medicine case near the bathroom sink."* Building such assistive agents is a problem of deep scientific and societal value.

To study this problem systematically, the embodied AI community has rallied around a problem called object-goal navigation (or ObjectNav) [1]. Given the name of an object (e.g., *"chair"*), ObjectNav involves exploring a 3D environment to find any instance of the specified object. The last few years have witnessed the development of new environments [2–5], annotated 3D scans [6, 7], datasets of human demonstrations [8], and approaches for ObjectNav [9–12], cumulatively leading to strong progress. For instance, the entries in the annual Habitat challenge [13] have jumped from 0% success (DD-PPO baseline in 2020) to 53% success (top entry in 2022 Habitat Challenge public leaderboard).

While this progress is exciting, we believe that a subtle but insidious assumption has snuck into this line of work: the closed-world assumption. We started by discussing an open-world scenario where a person may describe any object in language (e.g., *"screwdriver"*), but ObjectNav is currently formulated over a closed pre-determined vocabulary of object categories (*"chair"*, *"bed"*, *"sofa"*, etc.), with approaches using pre-trained object detectors and segmenters for these categories [8–11]. While this assumption may have been essential to get started on this problem, it is now important to move beyond it and ask – how can embodied agents find objects in an open-world setting?

In this work, we develop an approach for ObjectNav that is both *zero-shot*, i.e., does not require *any* ObjectNav annotations or training data, and *open-world*, i.e., does not require committing to a taxonomy of categories. Our key insight is that we can create a visiolinguistic embedding space to decouple two problems – (1) describing and representing semantic goals (*"chair"*, *"brown chair"*, picture of brown chair) from (2) learning to navigate to semantic goals.

Submitted to 36th Conference on Neural Information Processing Systems (NeurIPS 2022). Do not distribute.

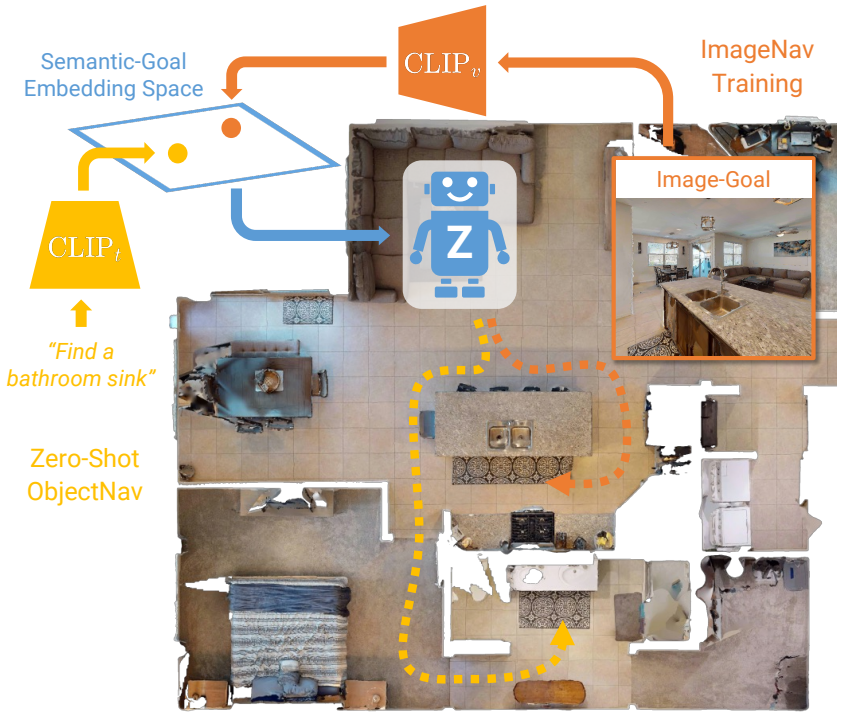

Figure 1: We propose projecting navigation goals (from images or text) into a common, semantic embedding space using a pre-trained vision and language model (CLIP), then learning semantic-goal navigation. Our approach enables *open-world* object-goal navigation in a *zero-shot* manner, allowing agents trained with image-goals to understand goals expressed in language (*"Find a bathroom sink."*).

To represent semantic goals (1), we leverage recent advances in multimodal AI research on learning a common embedding space for images and text using large collections of image-captions pairs. Specifically, we use CLIP [14], a method for training independent vision and language encoders that produce similar representations for paired data such as an image and its caption. As shown in Fig. 1, we use CLIP to transform image-goals (e.g., a picture of the kitchen island) and object-goals (e.g., *"bathroom sink"*) into *semantic-goals* representing navigation targets. Our main observation is that a semantic-goal produced from an image (e.g., a picture of the bathroom sink) should be similar to semantic goals produced from descriptions of the same target (e.g, *"bathroom sink"*). Thus, we hypothesize that these modalities (images and language) can be used interchangeably for creating semantic goals.

For learning to navigate to semantic goals (2), we validate our hypothesis. We train a semantic-goal navigation agent using image-goals transformed by CLIP. Then, we evaluate the learned navigation policy on `ObjectNav`, where goals are specified in language (e.g., *"chair"*). As a result, our agents perform `ObjectNav` without ever directly training for the task – i.e., in a zero-shot manner.

An important advantage of our approach is that it reduces the data labeling burden. Image-goals can be procedurally generated by randomly sampling points in 3D environments. This is in stark contrast to `ObjectNav`, which requires annotating each environment or collecting large-scale human demonstrations [8] for training. Secondly, the interface to our agents is a natural language description – matching the grand vision that inspires the `ObjectNav` task. Through this interface we can refine object-goals by, for instance, specifying object attributes (*"brown chair"*) or indicating which room the object is in (*"bathroom sink"*) – which is not possible with traditional `ObjectNav` agents.

We perform large-scale experiments on three `ObjectNav` datasets – Gibson [4], MP3D [7], and HM3D [15]. Our zero-shot agent (that has not seen a single 3D semantic annotation or `ObjectNav` training episode) achieves a 31.3% success in Gibson environments, which is a 20.0% absolute improvement over previous zero-shot results. In MP3D, our agent achieves 15.3% success, a 4.2% absolute gain over existing zero-shot methods. On HM3D, the SPL of our agent matches a state-of-the-art `ObjectNav` method [12] that trains with direct supervision from `ObjectNav` annotations.

Through ablations of our method, we discover that visual encoder pretraining has an outsized effect on zero-shot transfer – resulting in gains of 4.5% - 5.8% in success on the `ImageNav` training task, but improvements of 9.4% - 10.4% in downstream success on zero-shot `ObjectNav`. Similarly, increasing the number of training environments (from 72 to 1,000) leads to a small drop in `ImageNav` success, but results in a substantial improvement of 6.6% in success on zero-shot `ObjectNav`.

Finally, we qualitatively experiment with an open-world setting and observe that our `SemanticNav` agents can properly change behavior in response to instructions that include room information. For instance, when finding a *"bathroom sink"* the agent does not enter the kitchen, and when looking for a *"kitchen sink"* it does not enter bathrooms. Furthermore, we observe similar room awareness patterns for instructions such as *"Find a sink and a stove,"* where the target room (*"kitchen"*) can be inferred. Source code for reproducing our results will be publicly released upon acceptance.

2 Related Work

Image-Text Alignment Models. Recent progress in vision-and-language pretraining has led to models such as CLIP [14], ALIGN [16], and BASIC [17] that can perform open-world image classification, and achieve strong performance on standard computer vision benchmarks (e.g., ImageNet [18]). These models learn visual representations by training on massive datasets of image-caption pairs scraped from the web (e.g., the 400M pairs used for CLIP or 6.6B for BASIC). In this work, we take advantage of the semantic representations learned by CLIP to project navigation goals (e.g., *"brown chair"* or a picture of a chair) into a semantic-goal embedding space.

CLIP in a Navigation Agent. A straightforward approach for using CLIP in a visual navigation agent is to process the agent's observations and navigation instructions (e.g., *"Find a chair"*) with the CLIP image and text encoders, then learn a navigation policy that operates on these embeddings. Such a solution was explored in EmbCLIP [19] with promising results. However, this approach requires annotating training data to supervise the navigation policy, which is simply difficult and costly to collect at scale. As a result, existing training datasets tend to be small and agents generalize poorly to new settings. For instance, EmbCLIP only achieves an 8% success rate in finding objects that were not used in training. By contrast, we train using a transformed version of the image-goal navigation task, which does not require annotated environments. Thus, we are able to scale training to 1,000 unannotated 3D scenes, which substantially improves generalization (as demonstrated in Section 5).

Zero-Shot ObjectNav. Two recent works propose techniques for zero-shot `ObjectNav` [20, 21]. ZER [20] proposes a two-stage framework in which an image-goal navigation (`ImageNav`) agent is first trained from scratch. Then, independent encoders are trained to map from various modalities (including language) into the image-goal embedding space. A key challenge with this approach is that image-goal embeddings may not capture semantic information because semantic annotations are not used in `ImageNav` training. By contrast, our approach reverses these two stages, with CLIP pretraining representing stage one. Thus, our approach uses a goal embedding space that captures semantics by design. We empirically compare these proposed solutions directly in Section 5.

In concurrent work, CoW [21] uses a gradient-based visualization technique (GradCAM [22]) to perform object localization with CLIP. This is combined with a heuristic exploration policy to enable zero-shot object-goal navigation. In this paper, we demonstrate that learning a navigation policy using the scalable task of image-goal navigation leads to substantially better zero-shot object-goal navigation performance without using additional object localization techniques.

3 Preliminaries

Image-Text Alignment Models. Multimodal alignment models aim to learn a mapping from images v_i and text t_j into a shared embedding space such that representations for corresponding image-text pairs (e.g., a picture and its caption) are close (often measured by cosine similarity). Recent image-text alignment models [14, 16, 17] use a dual-encoder framework and optimize the InfoNCE [23] contrastive learning objective, which maximizes cosine similarity between representations of matching image-text pairs and minimizes similarity for non-matching pairs. Specifically, CLIP [14] was trained on 400M image-text pairs that cover a wide range of visual concepts.

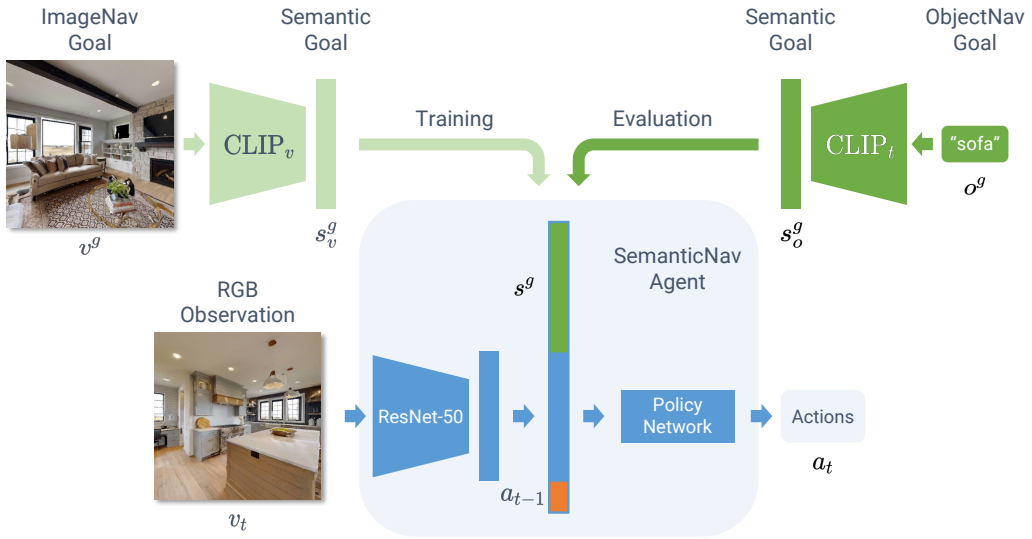

Figure 2: We create semantic goals for training by processing image-goals with a CLIP visual encoder. These goals are replaced with text embeddings for zero-shot `ObjectNav`. Our agent has a simple architecture: RGB observations are encoded with a pretrained ResNet-50, and a recurrent policy network predicts actions using encodings of the goal s^g, observation, and the previous action a_{t-1}.

CLIP can be used for *zero-shot* image classification on various tasks such as ImageNet [18] or HatefulMemes [24]. Here, zero-shot indicates that image classification labels are not used in training. Zero-shot classification is performed by selecting the class that maximizes the cosine similarity between the CLIP embedding of an image $\texttt{CLIP}_v(v_i)$ and CLIP text embeddings $\texttt{CLIP}_t(t_j)$ for a set of text prompts $\{t_0, t_1, \ldots, t_K\}$ designed for each of the K classes (e.g., *"a photo of a zebra"* or *"a hatespeech meme"*). In this work, we use the semantic embedding space of CLIP to represent navigation goals, which enables using free-form text for goal specification.

Image-Goal Navigation. In image-goal navigation (`ImageNav`) [25], agents explore an environment to find the location where a goal-image v^g was captured. We consider a setting in which both the goal-image and the agent's observations consist of RGB images taken from the agent's egocentric point of view. An agent can select from four actions: `MOVE_FORWARD` by 0.25m, `TURN_LEFT` by 30°, `TURN_RIGHT` by 30°, or `STOP`. The agent succeeds if it selects `STOP` within 1.0m of the goal.

An `ImageNav` episode is uniquely defined by a starting position and (reachable) goal viewpoint within a 3D environment. Thus, `ImageNav` training data can be procedurally generated without annotating the scene – i.e., the objects and rooms do not need to be labeled. As a result, the size of an `ImageNav` dataset is only limited by the number of environments available for training. In this work, we train agents for a variant of `ImageNav` that we term semantic-goal navigation (`SemanticNav`). By design, `SemanticNav` training data can also be procedurally generated without annotated environments.

4 Approach

This section describes our framework for training visual navigation agents. We use CLIP [14] to transform the image-goal navigation task into semantic-goal navigation. This allows training semantic-goal navigation agents at scale using HM3D [15]. At deployment, we use CLIP to convert object-goals (e.g., *"Find a chair"*) into semantic-goals, which enables *zero-shot* object-goal navigation. In other words, our agents execute object-goal navigation without ever directly training for the task.

4.1 Learning Semantic-Goal Navigation

We construct a semantic-goal navigation task (denoted here as `SemanticNav`) by encoding image-goals (e.g., an image of the bathroom sink) into the CLIP semantic embedding space using a CLIP visual encoder $\texttt{CLIP}_v$. Formally, given an image goal v^g, we generate a corresponding semantic goal $s_v^g = \texttt{CLIP}_v(v^g)$ (Fig. 2 top-left). The one-to-one mapping between goal images and semantic goals allows training `SemanticNav` agents using any `ImageNav` dataset. As in `ImageNav`, `SemanticNav` agents must locate the goal using egocentric RGB observations and an action space consisting of `MOVE_FORWARD`, `TURN_LEFT`, `TURN_RIGHT`, and `STOP`. However, in `SemanticNav`, the goal is only specified with the high-level goal vector s_v^g and agents never observe the underlying goal image v^g.

Conceptually, encoding image-goals with CLIP preserves semantic information about the goal, such as visual concepts that might be described in image captions (e.g., *"a sofa in a living room"*). However, semantic goals are less likely to include low-level features (e.g., the exact patterns in a wood floor) that do not correlate with web-scraped captions. While removing low-level information might make `SemanticNav` more difficult than `ImageNav`, our goal is to learn a navigation policy that transfers to `ObjectNav` in which agents only receives high-level goals (e.g., *"Find a sofa"*). As an added benefit, generating semantic goals as a pre-processing step substantially improves training time (by ∼3.5x).

Semantic-Goal Navigation Agent. Our agent architecture is shown in Fig. 2. At each timestep t, our agent receives an egocentric RGB observation v_t and a goal representation s_v^g. The observation is processed by a ResNet-50 [26] encoder, which is pretrained on the Omnidata Starter Dataset (OSD) [27] using self-supervised learning (DINO [28]) following the pretraining recipe presented in OVRL [12]. The output from the ResNet-50 encoder is concatenated with the goal representation s_v^g and an embedding of the agent's previous action a_{t-1} and then passed to the policy network composed of a two-layer LSTM. The policy network outputs a distribution over the action space.

We train our `SemanticNav` agent with reinforcement learning (RL). During RL training, we use two data augmentation techniques: color jitter and random translation (adapted from [12]). Specifically, we use DD-PPO [29] with a reward function proposed for `ImageNav` by Al-Halah et al. [20]:

$$r_t = r_{\text{success}} + r_{\text{angle-success}} - \Delta_{\text{dtg}} - \Delta_{\text{atg}} + r_{\text{slack}} \tag{1}$$

where $r_{\text{success}} = 5$ if the agent calls `STOP` within 1m of the goal (and 0 otherwise), $r_{\text{angle-success}} = 5$ if the agent is pointing with 25° of the goal heading when `STOP` is called (and 0 otherwise), Δ_{dtg} is the change in the agent's distance-to-goal, Δ_{atg} is the change in the agent's angle-to-goal but is set to 0 if agent is greater than 1m from the goal, and $r_{\text{slack}} = -0.01$ to encourage efficient navigation. In general, this reward function encourages both reaching the goal and looking towards the goal before calling `STOP`, which matches the requirements of the downstream `ObjectNav` task.

4.2 Zero-Shot Object-Goal Navigation

In `ObjectNav` [1], agents are given a target category (e.g., *"sofa"* or *"chair"*) and must locate any instance of that object (i.e., *"any sofa"* or *"any chair"*). Similar to `ImageNav`, `ObjectNav` requires exploring new environments that the agent has never seen before. However, in `ObjectNav`, the goal (e.g., *"sofa"*) provides a minimal amount of information about where the agent must go and it requires recognizing any version of the goal object in the new scene.

To address this task, we map an object-goal o^g (e.g., *"sofa"*) into the CLIP semantic embedding space using a CLIP text encoder $\texttt{CLIP}_t$ resulting in the semantic goal $s_o^g = \texttt{CLIP}_t(o^g)$. CLIP aligns image and text, thus the semantic goals from text s_o^g should be close (in terms of cosine similarity) to the CLIP visual embeddings s_v^g used in training. For zero-shot `ObjectNav` we do not use any prompt engineering (e.g., using a template such as "`A photo of a <>`"). Instead, we simply use the object name (e.g., *"sofa"*) as the object-goal input o^g.

5 Experimental Findings

This section experimentally studies training semantic-goal navigation agents and zero-shot transfer to `ObjectNav`. First, we evaluate in the traditional `ObjectNav` setting [1] where agents must find any instance of the goal object (*"Find a chair"*). Then, we explore variations of `ObjectNav` in which additional information, such as a room location (e.g., *"bathroom sink"*), is given to refine the task. These experiments aim to demonstrate both the effectiveness and versatility of our approach.

5.1 Experimental Setup

Training Dataset. We generate a `SemanticNav` dataset using the 1,000 training environments from HM3D [15]. First, we sample 3,000 `ImageNav` episodes for three difficulty levels corresponding with path length: EASY (1.5-3m), MEDIUM (3-5m), and HARD (5-10m). We follow the episode generation approach from [30]. This results in 9M navigation episodes for training. Next, we pre-process the goal-images with the ResNet-50 version of CLIP [14] to produce 1024 dimensional semantic goal vectors s_v^g for each navigation episode. During pre-processing, we further augment the dataset by sampling goal-images at four evenly-spaced heading angles to produce 36M total episodes for training. Sampling at multiple angles approximates the randomized sampling used in [20].

Agent Configurations. Two different agent configurations are frequently used in prior work on visual navigation. Configuration `A` is generally used for `ImageNav` and has an agent height of 1.5m, radius of 0.1m, and a single 128×128 RGB sensor with a 90° horizontal field-of-view (HFOV) placed 1.25m from the ground. Configuration `B` is typically used for `ObjectNav` and approximately matches a LoCoBot, with an agent height of 0.88m, radius of 0.18m, and a single 640×480 RGB sensor with a 79° HFOV placed 0.88m from the ground. Both configurations use the aforementioned step size of 0.25m and left and right turning angle of 30°.

Evaluation Datasets. We measure performance on one `ImageNav` and three `ObjectNav` datasets:

- `ImageNav` (Gibson) consists of 4,200 episodes from 14 Gibson [4] validation scenes. The dataset was produced by Mezghani et al. [30] for agents with configuration `A`.
- `ObjectNav` (Gibson) was generated by Al-Halah et al. [20] for agents with configuration `A`. The dataset consists of 1,000 episodes in 5 Gibson [4] validation scenes for 6 object categories.
- `ObjectNav` (HM3D), released with the Habitat 2022 challenge, consists of 2,000 episodes from 20 HM3D [15] validation scenes with objects from 6 categories, and uses agents with configuration `B`.
- `ObjectNav` (MP3D) released with the Habitat 2020 challenge, contains 2,195 episodes from 11 MP3D [7] validation scenes for 21 object categories, and requires agents with configuration `B`.

Due to the different agent configurations required by these evaluation datasets, we train agents with both settings to make fair comparisons with prior work on zero-shot `ObjectNav`. For all experiments, we report two standard metrics for visual navigation tasks: success rate (`SR`) and success rate weighted by normalized inverse path length (`SPL`) [31].

Implementation Details. We generate a `SemanticNav` dataset for each agent configuration (`A` and `B`). The CLIP ResNet-50 encoder processes 224 × 224 images. Accordingly, for configuration `A`, we render 512 × 512 RGB frames, then resize to 224 × 224. For configuration `B`, we render at 640 × 480, then resize and center crop. We train agents using PyTorch [32] and the Habitat simulator [2, 3]. Each training run was conducted on a single compute node with 8 NVIDIA A40 GPUs. We train agents for 500M steps, requiring ∼1,704 GPU-hours to train two agents (one for each configuration). Additional training hyperparamters are detailed in the Appendix. We report results using the best checkpoint, selected based on `ObjectNav` validation success rate (`SR`). During evaluations we sample actions from the agent's output distribution. We report results averaged over three evaluation runs.

Baselines. We compare results with two state-of-the-art zero-shot `ObjectNav` methods:

- **Zero Experience Required (ZER)** [20]: first trains an `ImageNav` agent composed of two ResNet-9 encoders for processing the goal-image and agent observations, and a policy network consisting of a 2-layer GRU. After training the navigation policy, a 2-layer MLP is trained to map from a goal object categories into the goal-image embedding space learned through `ImageNav` training. This mapping is learned using an in-domain dataset containing 14K images with object category labels.
- **CLIP on Wheels (CoW)** [21]: builds an occupancy map by projecting depth observations, then searches the environment with frontier-based exploration [33]. At each step, CoW calculates a 3D saliency map using a depth and RGB observations and the goal object category via Grad-CAM [22], a gradient-based visualization technique. When the 3D saliency exceeds a threshold the agent navigates to that location and stops. As such, CoW does not require a learned navigation policy.

We also compare with OVRL [12], a two-stage method for training visual navigation agents with *direct supervision*. In OVRL, a ResNet-50 encoder is pretrained with OSD [27] data using the

Table 1: **Zero-shot ObjectNav performance** on Gibson [4], HM3D [15], and MP3D [7] validation. All methods use a single RGB sensor for agent observations except CoW [21], which also uses depth observations and OVRL [12], which uses `GPS+Compass` for `ObjectNav`. Our approach (ZSON) substantially improves on previous zero-shot methods and narrows the gap to SOTA supervised methods such as OVRL [12], which is not zero-shot and provided for reference. We report ZSON results averaged over three evaluation trials, $\sigma = \{0.02\%, 0.46\%, 0.11\%\}$ for `ObjectNav` `SR` in Gibson, HM3D, and MP3D, respectively. **indicates reproduced results*

Method	ImageNav (Gibson)		ObjectNav (Gibson)	
	SPL	SR	SPL	SR
OVRL [12]	27.0%	54.2%	-	-
ZER [20]	21.6%	29.2%	-	11.3%
ZSON (ours)	**28.0%**	**36.9%**	12.0%	**31.3%**

(a) Configuration `A`

Method	ObjectNav (HM3D)		ObjectNav (MP3D)	
	SPL	SR	SPL	SR
OVRL [12]	12.3%*	32.8%*	7.0%	25.3%
CoW [21] (w/depth)	-	-	**6.3%**	11.1%
ZSON (ours)	12.6%	25.5%	4.8%	**15.3%**

(b) Configuration `B`

DINO [28] self-supervised learning algorithm. Then, the encoder is used in a navigation policy that is finetuned for either `ImageNav` or `ObjectNav` with task-specific training data. We use the OVRL pretraining stage in our approach. OVRL achieves state-of-the-art results on `ImageNav` and `ObjectNav` in our single RGB camera setting. We highlight OVRL in blue to indicate the use of direct supervision.

5.2 Zero-Shot Object-Goal Navigation

In Table 1 we study zero-shot `ObjectNav` performance. We compare with ZER [20] in Table 1a using a configuration `A` agent. We observe that our method (ZSON) results in a 7.7% absolute improvement in `SR` on `ImageNav`, which serves as a pretraining task for both ZER and our approach. On `ObjectNav`, we find that this translates to an even larger 20.0% absolute improvement in `SR`. Notably, our approach substantially reduces the gap between `ImageNav` pretraining and zero-shot `ObjectNav` from a difference in `SR` of 17.9% for ZER to only 5.6% with our approach. Accordingly, our approach recovers 85% of the pretraining success rate during zero-shot transfer to `ObjectNav`.

In Table 1b we compare with CoW [21] using a configuration `B` agent. On the MP3D `ObjectNav` validation set, we find that learning a navigation policy by training for `SemanticNav` improves `SR` by 4.2 absolute percentage points (a 38% relative gain). These results demonstrate that learning to navigate and (importantly) learning when to stop can improve zero-shot `ObjectNav` `SR` over the hand-designed navigation policy and stopping criteria proposed by CoW. Moreover, we expect further improvements in zero-shot `ObjectNav` performance from scaling our approach (e.g., by collecting more training environments). Such scaling is simply not possible with heuristic methods such as CoW because the navigation policy is not learned. The `SPL` of our approach is 1.5% lower than CoW. However, unlike CoW, our agent navigates without depth observations, which likely reduces path efficiency. On HM3D we find that our agent achieves a strong `SR` of 25.5% and `SPL` of 12.6%. Impressively, this zero-shot `SPL` matches OVRL [12], which is directly trained on human demonstrations [8] for the `ObjectNav` task (and does not use depth observations, as in our approach).

5.3 Ablation Results

In Table 2, we study the impact of two key design decisions within our method: (1) using a pretrained visual encoder and (2) using more environments for training. While pretraining a visual encoder is known to improve visual navigation task performance (demonstrated in [12]), here we study the impacts on zero-shot transfer to `ObjectNav`. We find that OVRL pretraining improves `ImageNav` success by 4.5% (rows 1 vs. 3) or 5.8% (rows 2 vs. 4) depending on the dataset used for training. However, the impact on zero-shot `ObjectNav` performance is substantially larger. Specifically, `ObjectNav` success improves by 9.4% (rows 1 vs. 3) and 10.4% (rows 2 vs. 4). These results suggest that a stronger visual encoder can improve zero-shot transfer to `ObjectNav`.

Table 2: **Ablations** of the visual encoder and 3D dataset used for training our `SemanticNav` agents.

			`ImageNav` (Gibson)		`ObjectNav` (Gibson)	
#	Encoder	Dataset	SPL	SR	SPL	SR
1	ResNet-9	Gibson	22.8%	33.3%	7.4%	15.3%
2	ResNet-9	HM3D	23.4%	31.1%	9.5%	20.9%
3	OVRL	Gibson	27.6%	**37.8%**	10.0%	24.7%
4	OVRL	HM3D	**28.0%**	36.9%	**12.0%**	**31.3%**

In rows 3 vs. 4, we switch the training dataset from the 72 Gibson [4] training environments (row 3) to the 1,000 (unannotated) HM3D [15] training environments. Surprisingly, we observe a 0.9% drop in `ImageNav` success, yet a 6.6% improvement in `ObjectNav` success (rows 3 vs. 4). A similar trend is observed in rows 1 vs. 2. These trends indicate that training environment diversity is particularly useful for zero-shot `ObjectNav`.

Table 3: **Comparison with ZER** [20] using a ResNet-9 and the Gibson dataset with our approach. Learning `SemanticNav` (Ours) outperforms learning `ImageNav` then language grounding (ZER [20]).

			`ImageNav` (Gibson)		`ObjectNav` (Gibson)	
Method	Visual Encoder	Training Dataset	SPL	SR	SPL	SR
ZER [20]	ResNet-9	Gibson	21.6%	29.2%	-	11.3%
Ours	ResNet-9	Gibson	**22.8%**	**33.3%**	**7.4%**	**15.3%**

5.4 Comparison with ZER.

In Table 3, we train our approach in Gibson environments without using a pretrained encoder. These settings match ZER [20], allowing for a direct comparison between the two methods. We observe that our approach results in a 4.0% improvement in zero-shot `ObjectNav` success (a 35% relative gain). These results demonstrate that our proposed approach of training for the semantic-goal navigation task outperforms the inverse approach, proposed by ZER, of first training for image-goal navigation and then learning a mapping from object categories into the image-goal embedding space.

5.5 Qualitative Analysis

In Fig. 3, we present qualitative examples of our agent navigating to more complex object descriptions (e.g., *"Find a bathroom sink"*). In each trial, the agent starts at the same position and heading (next to the front door looking into the house). The only thing that changes about the initial conditions is the instructions given to the agent (*"Find a..."* *"...bathroom sink"*, *"...kitchen sink"*, *"...sink and a toilet"*, or *"...sink and a stove"*). We run each instruction five times and highlight the first run in bold colors.

We find that given room information such as *"bathroom"* or *"kitchen"*, the agent appropriately finds a *"sink"* in the corresponding rooms in the house. Furthermore, in these examples the agent does not enter the *"kitchen"* when prompted to look for a *"bathroom sink,"* and vice-versa. In these long trajectories (ranging from 88 to 225 steps), we observe more exploration in the living room and direct navigation when target rooms are visible. We qualitatively observe interesting learned behaviors – for instance, the agent often performs a 360° turn before navigating, possibly to survey the environment.

Next, we experiment with variations in which room information can be inferred from the instruction, but is not explicit. We use *"sink and a toilet"* to indicate *"bathroom"* and *"sink and a stove"* for *"kitchen"*. In these examples, we discover that our agent still navigates to the correct rooms, suggesting that it learns some priors of indoor spaces, such as that a *"stove"* is often found within a *"kitchen."*

Figure 3: **Qualitative examples** for navigating to complex object descriptions. For each trail, the agent is spawned at the *start* position looking into the house (i.e., to the right on the maps) and given one of four instructions. Each instruction is run five times with the path for the first trail highlighted in bold colors. Our agent appropriately navigates to the correct rooms, demonstrating an understanding of both explicit (*"Find a kitchen sink"*) and implicit (*"Find a sink and a stove"*) room information.

6 Discussion

We present a zero-shot method for learning *open-world* object-goal navigation (`ObjectNav`). Our approach involves projecting image-goals into a semantic-goal embedding space using an image-and-text alignment model (CLIP). This creates a semantic-goal navigation task that does not require annotated 3D environments or collecting human demonstrations. Thus, our method is easy to scale. We discover that `SemanticNav` agents outperform previous zero-shot `ObjectNav` methods, and we identify two factors that have a strong impact on navigation success – pretraining the visual encoder and training in a diverse set of environments. In an open-world setting, we observe navigation patterns that suggest that `SemanticNav` agents can understand complex instructions, such as *"Find a sink and a stove."*

Limitations and Impact. `SemanticNav` agents appear to learn useful priors of indoor environments such as which room contains a *"stove."* However, agents may struggle in scenes where a navigation target is in an unusual location (e.g., a stove in a bedroom). Biases in the 3D environments used to train such agents might exaggerate these issues and affect deployments in non-traditional settings. Thus, interventions to mitigate this problem should be considered. Future work might explore how to use the natural language interface to `SemanticNav` agents to guide exploration in such scenarios.

References

[1] Dhruv Batra, Aaron Gokaslan, Aniruddha Kembhavi, Oleksandr Maksymets, Roozbeh Mottaghi, Manolis Savva, Alexander Toshev, and Erik Wijmans. Objectnav Revisited: On Evaluation of Embodied Agents Navigating to Objects. *arXiv preprint arXiv:2006.13171*, 2020. 1, 5

[2] Manolis Savva, Abhishek Kadian, Oleksandr Maksymets, Yili Zhao, Erik Wijmans, Bhavana Jain, Julian Straub, Jia Liu, Vladlen Koltun, Jitendra Malik, et al. Habitat: A platform for embodied ai research. In *ICCV*, 2019. 1, 6

[3] 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. *NeurIPS*, 2021. 6

[4] Fei Xia, Amir R Zamir, Zhiyang He, Alexander Sax, Jitendra Malik, and Silvio Savarese. Gibson Env: Real-World Perception for Embodied Agents. In *CVPR*, pages 9068–9079, 2018. 2, 6, 7, 8

[5] Eric Kolve, Roozbeh Mottaghi, Winson Han, Eli VanderBilt, Luca Weihs, Alvaro Herrasti, Daniel Gordon, Yuke Zhu, Abhinav Gupta, and Ali Farhadi. AI2-THOR: An Interactive 3D Environment for Visual AI. *arXiv*, 2017. 1

[6] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 1

[7] Angel Chang, Angela Dai, Thomas Funkhouser, Maciej Halber, Matthias Niessner, Manolis Savva, Shuran Song, Andy Zeng, and Yinda Zhang. Matterport3D: Learning from RGB-D Data in Indoor Environments. In *ThreeDV*, 2017. MatterPort3D dataset license: http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf. 1, 2, 6, 7

[8] Ram Ramrakhya, Eric Undersander, Dhruv Batra, and Abhishek Das. Habitat-web: Learning embodied object-search strategies from human demonstrations at scale. In *CVPR*, 2022. 1, 2, 7

[9] Devendra Singh Chaplot, Dhiraj Gandhi, Abhinav Gupta, and Ruslan Salakhutdinov. Object goal navigation using goal-oriented semantic exploration. In *NeurIPS*, 2020. 1

[10] Joel Ye, Dhruv Batra, Abhishek Das, and Erik Wijmans. Auxiliary tasks and exploration enable objectnav. In *ICCV*, 2021.

[11] Oleksandr Maksymets, Vincent Cartillier, Aaron Gokaslan, Erik Wijmans, Wojciech Galuba, Stefan Lee, and Dhruv Batra. Thda: Treasure hunt data augmentation for semantic navigation. In *ICCV*, 2021. 1

[12] 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. *arXiv preprint arXiv:2204.13226*, 2022. 1, 2, 5, 6, 7

[13] Karmesh Yadav, Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Oleksandr Maksymets, Rishabh Jain, Ram Ramrakhya, Angel X Chang, Alexander Clegg, Manolis Savva, Eric Undersander, Devendra Singh Chaplot, and Dhruv Batra. Habitat challenge 2022. https://aihabitat.org/challenge/2022/, 2022. 1

[14] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning Transferable Visual Models from Natural Language Supervision. In *ICML*, 2021. 2, 3, 4, 6

[15] Santhosh Kumar Ramakrishnan, Aaron Gokaslan, Erik Wijmans, Oleksandr Maksymets, Alexander Clegg, John M Turner, Eric Undersander, Wojciech Galuba, Andrew Westbury, Angel X Chang, Manolis Savva, Yili Zhao, and Dhruv Batra. Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI. In *NeurIPS Datasets and Benchmarks Track*, 2021. 2, 4, 6, 7, 8

[16] Chao Jia, Yinfei Yang, Ye Xia, Yi-Ting Chen, Zarana Parekh, Hieu Pham, Quoc Le, Yun-Hsuan Sung, Zhen Li, and Tom Duerig. Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision. In *ICML*, 2021. 3

[17] Hieu Pham, Zihang Dai, Golnaz Ghiasi, Hanxiao Liu, Adams Wei Yu, Minh-Thang Luong, Mingxing Tan, and Quoc V Le. Combined Scaling for Zero-shot Transfer Learning. *arXiv preprint arXiv:2111.10050*, 2021. 3

[18] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A Large-Scale Hierarchical Image Database. In *CVPR*, 2009. 3, 4

[19] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but Effective: CLIP Embeddings for Embodied AI. *arXiv preprint arXiv:2111.09888*, 2021. 3

[20] Ziad Al-Halah, Santhosh K Ramakrishnan, and Kristen Grauman. Zero Experience Required: Plug & Play Modular Transfer Learning for Semantic Visual Navigation. *arXiv preprint arXiv:2202.02440*, 2022. 3, 5, 6, 7, 8

[21] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. CLIP on Wheels: Zero-Shot Object Navigation as Object Localization and Exploration. *arXiv preprint arXiv:2203.10421*, 2022. 3, 6, 7

[22] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *ICCV*, 2017. 3, 6

[23] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 3

[24] Douwe Kiela, Hamed Firooz, Aravind Mohan, Vedanuj Goswami, Amanpreet Singh, Pratik Ringshia, and Davide Testuggine. The Hateful Memes Challenge: Detecting Hate Speech in Multimodal Memes. *NeurIPS*, 2020. 4

[25] Yuke Zhu, Roozbeh Mottaghi, Eric Kolve, Joseph J. Lim, Abhinav Kumar Gupta, Li Fei-Fei, and Ali Farhadi. Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning. *ICRA*, 2017. 4

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 5

[27] Ainaz Eftekhar, Alexander Sax, Jitendra Malik, and Amir Zamir. Omnidata: A Scalable Pipeline for Making Multi-Task Mid-Level Vision Datasets From 3D Scans. In *ICCV*, 2021. 5, 6

[28] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging Properties in Self-Supervised Vision Transformers. In *ICCV*, 2021. 5, 7

[29] 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. In *ICLR*, 2019. 5

[30] Lina Mezghani, Sainbayar Sukhbaatar, Thibaut Lavril, Oleksandr Maksymets, Dhruv Batra, Piotr Bojanowski, and Karteek Alahari. Memory-Augmented Reinforcement Learning for Image-Goal Navigation. *arXiv preprint arXiv:2101.05181*, 2021. 6

[31] Peter Anderson, Angel Chang, Devendra Singh Chaplot, Alexey Dosovitskiy, Saurabh Gupta, Vladlen Koltun, Jana Kosecka, Jitendra Malik, Roozbeh Mottaghi, Manolis Savva, et al. On Evaluation of Embodied Navigation Agents. *arXiv preprint arXiv:1807.06757*, 2018. 6

[32] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In *NeurIPS*, 2019. 6

[33] Brian Yamauchi. A frontier-based approach for autonomous exploration. In *CIRA*, 1997. 6

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [Yes]

(d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]

2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes]
   (b) Did you mention the license of the assets? [Yes]
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes]
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [Yes]
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes]
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]