ZERO-SHOT OBJECT-GOAL NAVIGATION USING MULTIMODAL GOAL EMBEDDINGS

A Dissertation
Presented to
The Academic Faculty

By

Gunjan Aggarwal

In Partial Fulfillment
of the Requirements for the Degree
Master of Science in the
College of Computing
Department of Computer Science

Georgia Institute of Technology

May 2023

ZERO-SHOT OBJECT-GOAL NAVIGATION USING MULTIMODAL GOAL EMBEDDINGS

Thesis committee:

Dr. Dhruv Batra

Georgia Institute of Technology & FAIR, Meta

Dr. Judy Hoffman

Georgia Institute of Technology

Dr. Devi Parikh

Georgia Institute of Technology & Generative AI, Meta

Date approved: April 21, 2023

What we learn with pleasure we never forget.

Alfred Mercier

To my sister Chetna Aggarwal.

ACKNOWLEDGMENTS

It was the most rewarding and enriching experience of my life so far.

To Dhruv Batra, my thesis advisor and the person behind all the confidence I have in myself. I have been immensely amazed by his in-depth knowledge of the ML domain and his prowess of making connection between distinct concepts and piecing everything together. Dhruv is also an amazing teacher - his *Deep Learning* class was the key to landing an internship and thereafter a job offer in these stressful times. I would like to thank Dhruv for his support throughout my Masters.

To Devi Parikh, for giving me the opportunity to conduct academic research and boosting my career path. Her exceptional attention to detail, ability to articulate thoughts, and prompt responses have significantly contributed to my growth as a researcher. Devi has been a constant source of support throughout my Master's journey, providing guidance and assistance whenever I felt lost, both academically and personally. Working directly with her in the field of Creative AI was an absolute privilege. From brainstorming ideas to project implementation, from steering the project in the right direction to writing, submission, website creation, and ultimately presentation, the entire experience was a valuable learning opportunity for me. Devi is not only an inspiration to me as a researcher, but also as a person, making her my favorite *Human of AI*. I am truly grateful for everything she has done for me.

To Judy Hoffmann, for providing invaluable inputs and helping me build a strong foundation for my thesis project. Additionally, her course, *Machine Learning with Limited Supervision* in Fall 2021, was my favorite from my Master's program. This course helped me develop the essential skill of reading and comprehending research papers, which proved to be immensely beneficial for my research projects later on.

I am grateful to Devi and Dhruv for giving me the golden opportunity to attend CVPR, my first ever in-person conference and NeurIPS, my first in-person conference as a presenter. I would also like to thank them for having annual lab retreats that allowed me to form stronger connections with my lab mates.

To Abhishek Sinha, for being a wonderful mentor during my first Deep Learning project and guiding me throughtout my first submission to a conference workshop. He made settling in a new continent less daunting. Today, I would not be where I am without him.

I would like to thank each and every member of the Computer Vision Machine Learning & Perception (CVMLP) Lab for providing a lovely atmosphere and constructive feedback during lab presentations during my Masters. This includes – Ram Ramrakhya, Joanne Truong, Naoki Yokoyama, Karmesh Yadav, Harsh Agrawal, Samyak Datta, Erik Wijmans, Nirbhay Modhe, Meera Hahn, Andrew Szot, Gunjan Chhablani, Harsh Maheshwari, Mukul Khanna and Pranav Putta. Arjun Majumdar - for being a great mentor and collaborator, Sameer Dharur - for being available through critical moments during my Masters, Ramprasaath Selvaraju - for providing timely career advice towards the end of my degree.

My special thanks to Ram Ramrakhya - from being my go-to person for silliest doubts, to cooking amazing Indian food, he has been a dear friend and a constant support system at Georgia Tech. I would like to thank him for making my Masters life better in every aspect.

I would like to thank Mayank Lunayach, for all the positivity, encouragement and constantly rooting for me through past 2 years. Thank you for educating me about football and preparing me to watch FIFA World Cup 2026 in USA.

I would also like to thank Soumajit Sen, for being there since the start, celebrating all the achievements and lifting me up during failures. To Abhineet, Ashwin, Mukund, Mayank, Pradeepti, Soumajit, Swapnil and Tanya for the laughter, dance sessions, board games sessions and all the fun memories during these past 2 years.

To my grand-parents Sita Ram Aggarwal and Gita Aggarwal, for every-day morning video calls, helping my happily kick-start the day. To my mother, Sunita Aggarwal, for her relentless sacrifices over the years, for her unconditional love and support. To my brother, Ashish Aggarwal, for all the funny memes exchange and light conversations to keep me sane. To our lovely pet Snowie, just for existing in our lives.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

My thesis presents a scalable approach for learning *open-world* object-goal navigation (`ObjectNav`) – the task of asking a virtual robot (agent) to find any instance of an object in an unexplored environment (e.g., *"find a sink"*). The approach is entirely *zero-shot* – i.e., it does not require `ObjectNav` rewards or demonstrations of any kind. Instead, we train on the image-goal navigation (`ImageNav`) task, in which agents find the location where a picture (i.e., goal image) was captured. Specifically, we encode goal images into a multimodal, semantic embedding space to enable training semantic-goal navigation (`SemanticNav`) agents at scale in unannotated 3D environments (e.g., HM3D). After training, `SemanticNav` agents can be instructed to find objects described in free-form natural language (e.g., *"sink," "bathroom sink,"* etc.) by projecting language goals into the same multimodal, semantic embedding space. As a result, our approach enables 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. For reference, these gains are similar or better than the 5% improvement in success between the Habitat 2020 and 2021 `ObjectNav` challenge winners. 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"*).

Publications

- **ZSON: Zero-Shot Object-Goal Navigation using Multimodal Goal Embeddings**
  Gunjan Aggarwal*, Arjun Majumdar*, Bhavika Devnani, Judy Hoffman, Dhruv Batra
  The Thirty-Sixth Annual Conference on Neural Information Processing Systems (NeurIPS 2022).

CHAPTER 1
INTRODUCTION

Imagine asking a home assistant robot to find 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 (`ObjectNav`) [1]. Given the name of an object (e.g., *"chair"*), `ObjectNav` involves exploring a 3D environment to find any instance of the object. The last few years have witnessed the development of new environments [2, 3, 4, 5, 6], annotated 3D scans [7, 8, 9], datasets of human demonstrations [10], and approaches for ObjectNav [11, 12, 13, 14, 15, 16], cumulatively leading to strong progress. For instance, the entries in the annual Habitat challenge [17] have jumped from 6% success (DD-PPO baseline in 2020) to 53% success (top entry in ongoing 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., *"flat-head screwdriver"*), but `ObjectNav` is currently formulated over a closed predetermined vocabulary of object categories (*"chair"*, *"bed"*, *"sofa"*, etc.), with approaches using pre-trained object detectors and segmenters for these categories [10, 11, 12, 13]. 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` rewards or demonstrations, and *open-world*, i.e., does not require committing to a taxonomy of categories. Our key insight is that we can create a

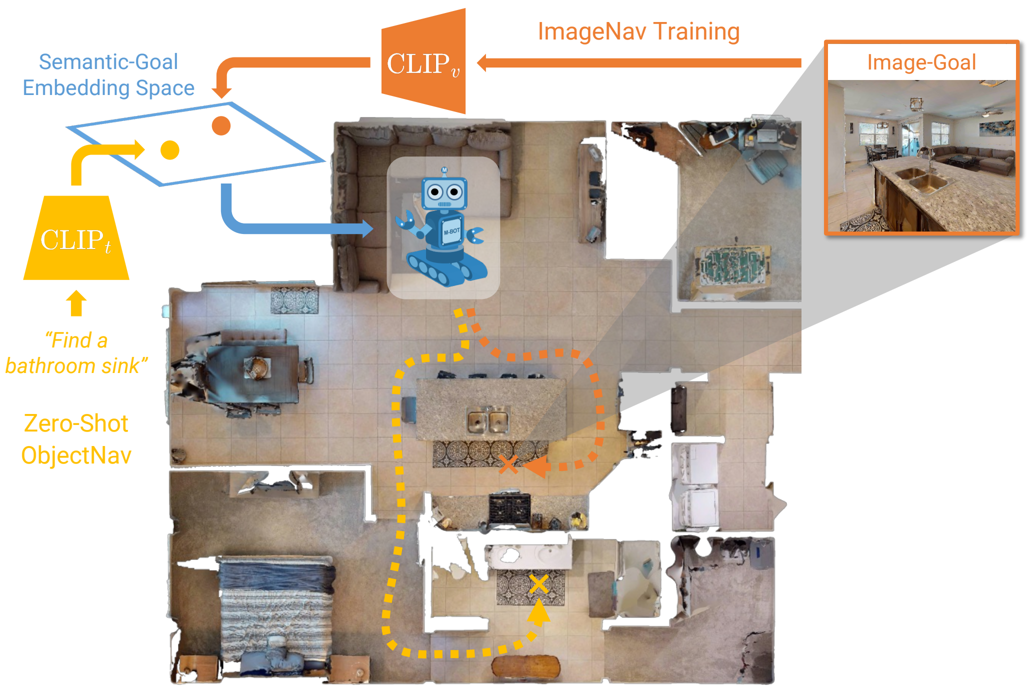

Figure 1.1: We propose projecting navigation goals (from images or text) into a common, semantic embedding space using a pre-trained vision and language model (CLIP). This allows agents trained with image-goals to understand goals expressed in free-form natural language (e.g., *"Find a bathroom sink."*). Accordingly, our approach enables *open-world* object-goal navigation in a *zero-shot* manner – i.e., without using ObjectNav rewards or demonstrations for training.

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.

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 [18], a method for training dual vision and language encoders that produce similar representations for paired data such as an image and its caption. As shown in Figure 1.1, 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.

Accordingly, for learning to navigate to semantic goals (2), we train agents using image-goals encoded via CLIP's image encoder. Then, we evaluate the learned navigation policy on `ObjectNav`, where goals are specified in language (e.g., *"chair"*) and encoded via CLIP's text encoder. 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 3D meshes [7, 8, 9] and potentially collecting large-scale human demonstrations [10] for training. Secondly, the interface to our agents is a natural language description – matching the grand vision that inspired 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 [8], and HM3D [19]. 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 [20]. In MP3D, our agent achieves 15.3% success, a 4.2% absolute gain over existing zero-shot methods[21]. For reference, these gains are on par or better than the 5% improvement in success between the Habitat 2020 and 2021 `ObjectNav` challenge winners. On HM3D, our agent's zero-shot SPL matches a state-of-the-art `ObjectNav` method [16] that trains with direct supervision from 40k human demonstrations.

Additionally, we study two techniques that are used in our approach to improve zero-shot `ObjectNav` performance. First, we find that pretraining the visual observation encoder has an outsized effect on zero-shot transfer. Specifically, success on the `ImageNav` training task improves 4.5% - 5.8%, while downstream success on zero-shot `ObjectNav` improves by 9.4% - 10.4%. Similarly, increasing the number of training environments (from 72 to 800) 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.

CHAPTER 2
RELATED WORK

This thesis builds on research studying image-text alignment techniques (e.g., CLIP [18]) and their use in visual navigation. In this section, we discuss methods most related to our proposed approach.

2.1 Image-Text Alignment Models.

Recent progress in vision-and-language pretraining has led to models such as CLIP [18], ALIGN [22], and BASIC [23] that can perform open-world image classification, and achieve strong performance on standard computer vision benchmarks (e.g., ImageNet [24]). 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., a picture of a brown chair or *"brown chair"*) into a multimodal, semantic-goal embedding space.

2.2 CLIP for Visual Navigation.

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 [25] with promising results. However, this approach requires ObjectNav rewards or demonstrations to supervise the navigation policy, which is 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 the image-goal navigation task, which does not require annotated environments. Thus, we are able to scale training to 800 unannotated 3D scenes, which substantially improves generalization (as demonstrated in **??**).

2.3 Zero-Shot ObjectNav.

Two recent works [20, 21] directly address our motivation (zero-shot `ObjectNav`) and are most related. First, 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. Instead, an `ImageNav` agent trained from scratch may learn to pattern match visual observations and goal image embeddings. 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 demonstrate the benefits of our proposed approach in **??**.

In concurrent work, CLIP-on-Wheels (CoW) [21] uses a gradient-based visualization technique (GradCAM [26]) with CLIP to localize objects in the agent's observations. This is combined with a heuristic exploration policy to enable zero-shot object-goal navigation. In contrast, we demonstrate that learning a navigation policy can substantially outperform the heuristic exploration approach proposed in [21] without using explicit object localization techniques.

CHAPTER 3
APPROACH

3.1 Preliminaries: Image-Text Alignment and Image-Goal Navigation

3.1.1 Image-Text Alignment Models.

Multimodal alignment models aim to learn a mapping from images v and text t into a shared embedding space such that representations for corresponding image-text pairs (e.g., a picture and its caption) are similar. Recent image-text alignment models [18, 22, 23] use a dual-encoder framework and optimize the InfoNCE [27] contrastive learning objective, which maximizes cosine similarity between representations of matching image-text pairs and minimizes similarity for non-matching pairs. In this work, we leverage CLIP [18], which was trained on 400M image-text pairs that cover a wide range of visual concepts.

3.1.2 Image-Goal Navigation.

In image-goal navigation (`ImageNav`) [28], agents explore an environment to find the position 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 use `ImageNav` to train visual navigation agents at

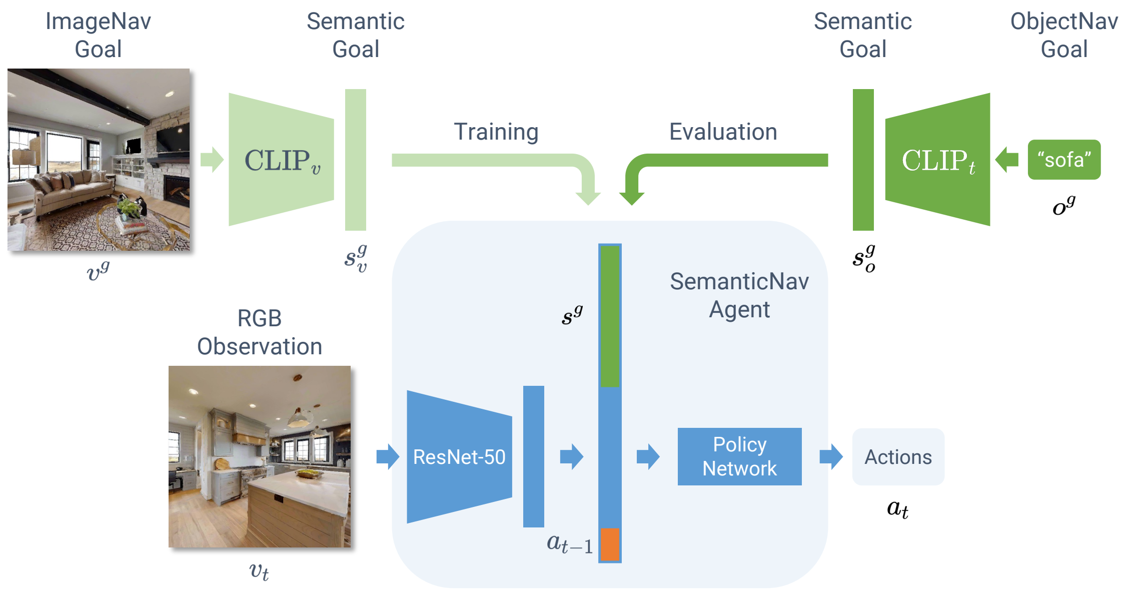

Figure 3.1: We tackle both `ImageNav` and `ObjectNav` via a common `SemanticNav` agent. This agent accepts a semantic goal embedding (s^g), which comes from either CLIP's visual encoder ($\mathtt{CLIP}_v$) in ImageNav or CLIP's textual encoder ($\mathtt{CLIP}_t$) in ObjectNav. Our agent has a simple 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}.

scale (in terms of the number of training environments).

3.2 Training Framework

This section describes our framework for training visual navigation agents. We use CLIP [18] to produce semantic goal embeddings of image-goals (e.g., a picture of the sink) and object-goals (e.g., *"sink"*). This allows training semantic-goal navigation agents at scale using image-goals in HM3D environments [19], then deploying these agents for object-goal navigation in a *zero-shot* manner. In other words, our agents execute object-goal navigation without ever directly training for the task.

3.2.1 Learning Semantic-Goal Navigation

As illustrated in Figure 3.1 (top-left), given an image-goal v^g, we use a CLIP visual encoder $\texttt{CLIP}_v$ to generate a semantic goal embedding $s_v^g = \texttt{CLIP}_v(v^g)$ that is used to guide navigation. 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 goal embeddings 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 the navigation task more difficult, our goal is to learn a policy that transfers to `ObjectNav` in which agents only receives high-level goals (e.g., *"Find a sofa"*). As an added benefit, generating semantic goal embeddings as a pre-processing step substantially improves training time (by ∼3.5x).

Our agent architecture is shown in Figure 3.1. 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 [29] encoder, which is pretrained on the Omnidata Starter Dataset (OSD) [30] using self-supervised learning (DINO [31]) following the pretraining recipe presented in OVRL [16]. 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 [16]). Specifically, we train with DD-PPO [32] using a reward function proposed for `ImageNav` by [20]:

$$r_t = r_{\text{success}} + r_{\text{angle-success}} - \Delta_{\text{dtg}} - \Delta_{\text{atg}} + r_{\text{slack}} \tag{3.1}$$

where $r_{\text{success}} = 5$ if `STOP` is called when the agent is within 1m of the goal position (and

0 otherwise), $r_{\text{angle-success}} = 5$ if `STOP` is called when the agent is within 1m of the goal position and the agent is pointing within 25° of the goal heading – i.e., the direction the camera was pointing when the goal image was collected – (and 0 otherwise), Δ_{dtg} is the change in the agent's distance-to-goal – i.e., the geodesic distance to the goal position, Δ_{atg} is the change in the agent's angle-to-goal – i.e., the difference between the agent's heading and the goal heading – but is set to 0 if the 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.

3.2.2 Zero-Shot Object-Goal Navigation

Recall that 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 transform object-goals o^g (e.g., *"sofa"*) into semantic goal embeddings using the CLIP text encoder $\texttt{CLIP}_t$, which results 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. To keep our approach simple and easily reproducible, 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.

CHAPTER 4
EXPERIMENTS

This section studies the zero-shot `ObjectNav` performance of our proposed approach. First, we evaluate our method 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.

4.1 Experimental Setup

4.1.1 Training Dataset.

We generate a dataset for training our `SemanticNav` agent using the 800 training environments from HM3D [19]. First, we sample 9k `ImageNav` episodes for each HM3D scan, split equally between 3 difficulty levels corresponding with path length: EASY (1.5-3m), MEDIUM (3-5m), and HARD (5-10m). We follow the episode generation approach from [33]. This results in 9k $\times$ $800 = 7.2$M navigation episodes for training. Next, we pre-process the goal-images with the ResNet-50 version of CLIP [18] 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].

4.1.2 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°.

4.1.3 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 [33] for agents with configuration `A`.

– `ObjectNav` (Gibson) was generated by [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 [19] 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 [8] 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) [34].

4.1.4 Implementation Details.

We generate a SemanticNav dataset for each agent configuration (A and B). The CLIP ResNet-50 encoder processes 224 $\times$ 224 images. Accordingly, for configuration A, we render 512 $\times$ 512 RGB frames, then resize to 224 $\times$ 224. For configuration B, we render at 640 $\times$ 480, then resize and center crop. We train agents using PyTorch [35] 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 $\sim$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.

4.1.5 Baselines.

We provide comparisons with the, to the best of our knowledge, only two existing zero-shot methods for object-goal navigation (ObjectNav):

– **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 [36]. At each step, CoW calculates a 3D saliency map using a depth and RGB observations and the goal object category via Grad-CAM [26], 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.

4.1.6 Fully-Supervised ObjectNav.

To understand the gap to fully-supervised `ObjectNav` methods, we compare with OVRL [16], a two-stage framework that achieves state-of-the-art `ObjectNav` results in our single RGB camera setting. We highlight OVRL in blue to indicate the use of direct supervision.

CHAPTER 5
RESULTS

5.1 Zero-Shot Object-Goal Navigation

In Table 5.1 we report zero-shot `ObjectNav` performance. We compare with ZER [20] in Table 5.1a using agent configuration `A`. Notice that our agent is stronger than ZER on `ImageNav`, which is the base pretraining task before ObjectNav can be studied. Specifically, we observe a 7.7% improvement in `ImageNav` `SR` (29.2% $\rightarrow$ 36.9%). This improvement results from (1) learning to navigate to semantic goal embeddings (as proposed in this work) instead of navigating to image-goal embeddings that are learned from scratch (as done in ZER), (2) using more diverse training environments, and (3) from using a pretrained visual encoder. We provide additional comparisons with ZER using the same set of training environments and without using visual encoder pretraining in Section 5.2, where we also observe improved performance. In Table 5.1a, we see even larger improvements in `ObjectNav` `SR` of 20.0% (11.3% $\rightarrow$ 31.3%). These results indicate that our design decisions are particularly useful for zero-shot `ObjectNav`.

In Table 5.1b we compare with CoW [21] using agent configuration `B`. In `ObjectNav` on the MP3D validation set, we find that training a `SemanticNav` agent improves `ObjectNav` `SR` by 4.2% absolute and 37.8% relative (11.1% $\rightarrow$ 15.3%). These results demonstrate that learning a navigation policy improves zero-shot `ObjectNav` `SR` over the hand-designed exploration strategy 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

Table 5.1: **Zero-shot ObjectNav performance** on Gibson [4], HM3D [19], and MP3D [8] validation. All methods use a single RGB sensor for agent observations except CoW [21], which also uses depth observations and OVRL [16], which uses `GPS+Compass` for `ObjectNav`. Our approach (ZSON) substantially improves on previous zero-shot methods and narrows the gap to SOTA fully-supervised methods such as OVRL [16], which is not zero-shot and provided for reference. We report ZSON results averaged over three evaluation trials. The standard deviation in ZSON `ObjectNav` `SR` is 0.02% in Gibson, 0.46% in HM3D, and 0.11% in MP3D. **indicates reproduced results*

	`ImageNav` (Gibson)		`ObjectNav` (Gibson)	
Method	`SPL`	`SR`	`SPL`	`SR`
OVRL [16]	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`

	`ObjectNav` (HM3D)		`ObjectNav` (MP3D)	
Method	`SPL`	`SR`	`SPL`	`SR`
OVRL [16]	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`

without depth observations, which may reduce 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 [16], which is directly trained on 40k human demonstrations [10] for the `ObjectNav` task with imitation learning.

5.2 Comparison with ZER without Encoder Pretraining and Training Environment Diversity

In Table 5.2, we train our approach in Gibson environments (instead of HM3D) and do not use a pretrained observation 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% absolute and 35% relative improvement in zero-shot `ObjectNav` success (11.3% $\rightarrow$ 15.3%). These results demonstrate that learning to navigate to semantic-goal embeddings outperforms the inverse approach proposed by ZER of first training for image-goal navigation, then learning a mapping from object categories into the image-goal embedding space.

Table 5.2: **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]).

Method	Visual Encoder	Training Dataset	`ImageNav` (Gibson)		`ObjectNav` (Gibson)	
			`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%**

5.3 Additional Ablations

In Table 5.3, we study the impact of two key design decisions within our method: (1) the visual observation encoder and (2) the number of training environments. While pretraining the visual observation encoder is known to improve visual navigation task performance (demonstrated in [16]), 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 strong visual encoder is often essential for zero-shot transfer to `ObjectNav`.

In rows 3 vs. 4, we switch the training dataset from the 72 Gibson [4] training environments (row 3) to the 800 (unannotated) HM3D [19] 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`.

5.3.1 Qualitative Analysis

In Figure 5.1, 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

Table 5.3: **Ablations** of the visual encoder and dataset used for training our `SemanticNav` agents.

#	Visual Encoder	Training Dataset	`ImageNav` (Gibson) `SPL`	`ImageNav` (Gibson) `SR`	`ObjectNav` (Gibson) `SPL`	`ObjectNav` (Gibson) `SR`
1	ResNet-9 from scratch	Gibson	22.8%	33.3%	7.4%	15.3%
2	ResNet-9 from scratch	HM3D	23.4%	31.1%	9.5%	20.9%
3	OVRL (ResNet-50, pretrained)	Gibson	27.6%	**37.8%**	10.0%	24.7%
4	OVRL (ResNet-50, pretrained)	HM3D	**28.0%**	36.9%	**12.0%**	**31.3%**

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"*). Since the agent's policy is stochastic, we show 5 sampled rollouts 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."*

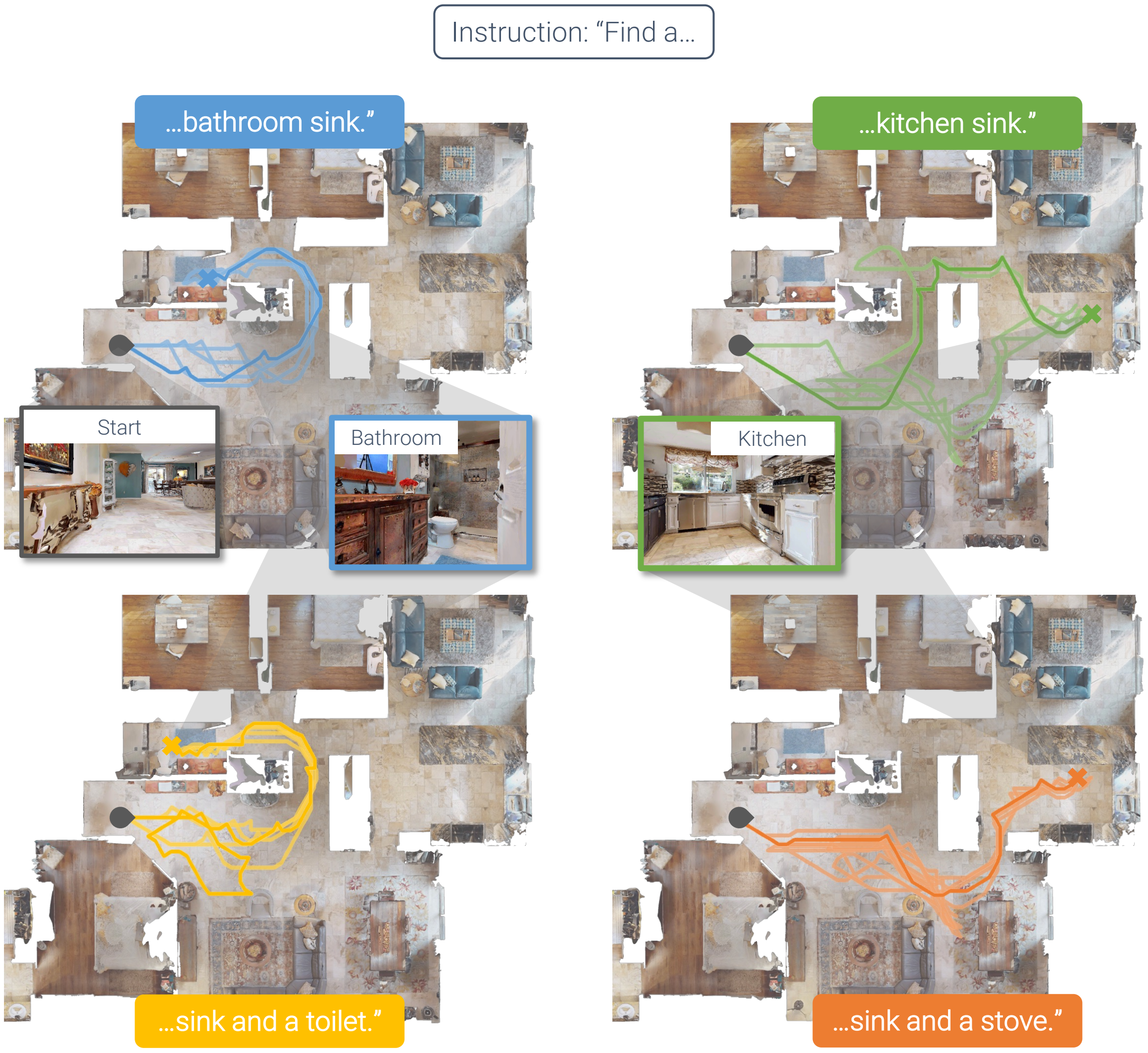

Figure 5.1: **Qualitative examples** for navigating to complex object descriptions. For each trail, the agent is spawned at the *start* position looking into the house (i.e., to the right on the maps) and given one of four instructions. Each instruction is run five times with the path for the first trail 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.

CHAPTER 6
DISCUSSION AND FUTURE WORK

This thesis presents 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.

Future Work. While our evaluation of `SemanticNav` agents on Habitat is a step towards assessing performance to novel objects, ultimately, real-world performance matters the most. Hence, a large-scale benchmarking of open-world object-navigation agents in simulation and the real-world is one potential direction to explore. Additionally, these agents might struggle in environments where navigation targets are out of place (e.g., a toy on the coffee table), which is often the case in real-world, generalizing to such instances of objects is another step towards building open-vocab `ObjectNav` agents.

Appendices

APPENDIX A

ADDITIONAL DETAILS AND RESULTS

Table A.1: **Comparison of ObjectNav methods.** Open-world methods are not limited to a closed set of object categories. Zero-shot methods do not use `ObjectNav` annotations for training.

Method	Open-World	Zero-Shot
Fully-Supervised Methods [10, 11, 12, 13, 16]	✗	✗
EmbCLIP [25]	✓	✗
ZER [20]	✗	✓
CoW [21]	✓	✓
ZSON (ours)	✓	✓

A.1 Extended Discussion of Related Work

In Table A.1, we compare object-goal navigation (`ObjectNav`) methods along two dimensions: *open-world* and *zero-shot*. Open-world methods are not restricted to object categories from a closed predetermined vocabulary (e.g., *"chair"*, *"bed"*, *"sofa"*, etc.). Zero-shot methods do not use `ObjectNav` annotations (e.g., labeled environments or large-scale human demonstrations [10]) for training.

Traditional, fully-supervised `ObjectNav` methods [10, 11, 12, 13, 16] rely on a closed-world assumption and task-specific training data – i.e., they are neither open-world nor zero-shot. EmbCLIP [25] can be used for open-world `ObjectNav` because it pre-processes object-goals (e.g., *"chair"*) with a CLIP [18] text encoder CLIP$_t$. Thus, the EmbCLIP interface allows describing objects using the open-vocabulary supported by CLIP$_t$. However, EmbCLIP is trained directly with `ObjectNav` annotations based on labeled 3D environments. Consequently, EmbCLIP is not a zero-shot method.

The method proposed in [20] (ZER) is zero-shot because it uses the image-goal navigation (`ImageNav`) task for training. However, ZER cannot perform open-world `ObjectNav` because it relies on a mapping from a closed-set of object categories into the image-goal embedding space for transfer to object-goal navigation. CoW [21] is a zero-shot method that does not require training a navigation policy. Instead, CoW uses a heuristically defined policy that has no ability to learn about indoor layouts of home environments (e.g., the fact that *"stoves"* are found in *"kitchens"* as illustrated in Fig. 5.1). However, CoW is able to perform open-world `ObjectNav` through the use of CLIP visual and text encoders.

By contrast, our approach (ZSON) uses CLIP to project image-goals and object-goals into a common semantic-goal embedding space, which converts `ImageNav` and `ObjectNav` into semantic-goal navigation (`SemanticNav`). This enables training with semantic-goals derived from images, followed by *zero-shot* transfer to *open-world* `ObjectNav`.

An interesting direction for future work might be to train `SemanticNav` agents with multi-task training using semantic-goals derived from both image- and object-goals. Such a solution would not be zero-shot. However, it might combined the advantages of large-scale training with image-goals (as used in our approach) with the advantages of smaller-scale task-specific training with object-goals (as used in EmbCLIP). We present initial results in this direction in Section A.2.

Table A.2: **Results of finetuning** with ObjectNav annotations. **indicates reproduced results*

#	Method	Dataset	SPL	SR
1	OVRL [16]	MP3D	7.0%	**25.3%**
2	ZSON (ours)	MP3D	4.8%	15.3%
3	ZSON finetuned 25M steps (ours)	MP3D	**9.2%**	22.9%
4	OVRL [16]	HM3D	12.3%*	32.8%*
5	ZSON (ours)	HM3D	12.6%	25.5%
6	ZSON finetuned 100M steps (ours)	HM3D	**27.0%**	**49.6%**

A.2 Results of ObjectNav Finetuning

In this section, we study the benefits of additional task-specific training by finetuning ZSON agents using `ObjectNav` annotations – i.e., manually labeled training environments. Specifically, we initialize with a `SemanticNav` agent trained for 500M steps of experience in HM3D environments using image-goals. Then, we finetune for ObjectNav in either MP3D [8] or HM3D [19] annotated environments using reinforcement learning (RL) with the finetuning approach from [37].

In Table A.2, we find that finetuning ZSON agents results in 7.6% - 24.1% absolute improvements in `ObjectNav` success rates (SR). Specifically, in row 3 we finetune for 25M steps in MP3D environments. This leads to a 7.6% absolute improvement in SR (15.3% $\rightarrow$ 22.9%) and a 4.4% absolute improvement in SPL (4.8% $\rightarrow$ 9.2%). This 9.2% SPL surpasses the state-of-the-art in the RGB-only setting of 7.0% SPL, which was set by OVRL [16] (row 1) using direct supervision from 40k human demonstrations in MP3D environments. Similarly, in row 6, we finetune for 100M steps in HM3D environments. This results in a 24.1% absolute improvement in SR (25.5% $\rightarrow$ 49.6%) and a 14.4% absolute improvement in SPL (12.6% $\rightarrow$ 27.0%). These results exceed the OVRL [16] baseline presented in row 4 (which was trained in MP3D environments and is identical to the agent in row 1) by 16.8% absolute in SR (32.8% vs. 49.6%) and 14.7% absolute in SPL (12.3% vs. 27.0%).

Table A.3: **Additional ablations** of the visual encoders used for training our `SemanticNav` agents.

			`ImageNav` (Gibson)		`ObjectNav` (Gibson)	
#	Encoder	Dataset	`SPL`	`SR`	`SPL`	`SR`
1	ResNet-9 (scratch)	HM3D	23.4%	31.1%	9.5%	20.9%
2	ResNet-50 (scratch)	HM3D	22.4%	28.2%	6.9%	16.6%
3	ResNet-50 (OVRL)	HM3D	**28.0%**	**36.9%**	**12.0%**	**31.3%**

A.3 Additional Ablation Experiments

Table 5.3 contains ablations of the visual encoder used to process RGB observations within our agent. In Table A.3, we provide additional ablations that compare the ResNet-9 and ResNet-50 architectures with and without pretraining – i.e., from scratch vs. using OVRL [16]. We find that without pretraining, switching to the larger ResNet-50 encoder leads to a 2.9% drop in `ImageNav SR` and a 4.3% drop in `ObjectNav SR` (rows 1 vs. 2). By contrast, using OVRL pretraining, performance improves on all tasks across all metrics. For instance, `ImageNav SR` improves by 5.8% and `ObjectNav SR` improves by 10.4% (rows 1 vs. 3).

Table A.4: **Hyperparameters** used to train `SemanticNav` agents.

Parameter	Value
Number of GPUs	8
Number of environments per GPU	32
Rollout length	64
PPO epochs	2
Number of mini-batches per epoch	2
Optimizer	Adam
Learning rate	2.25×10^{-4}
Weight decay	1.0×10^{-6}
Epsilon	1.0×10^{-5}
PPO clip	0.2
Generalized advantage estimation	True
γ	0.99
τ	0.95
Value loss coefficient	0.5
Max gradient norm	0.2
DDPPO sync fraction	0.6

A.4 Additional Training Details

Table A.4 details the default hyperparameters used in all of our training runs. For ablation experiments in Gibson [4] environments we reduce the "number of environments per GPU" to 28 for ResNet-9 experiments and 20 for ResNet-50 due to GPU memory constraints. We use the ResNet-9 implementation from [20] and ResNet-50 implementation from [2].

A.5 Additional Qualitative Results

In Figs. A.1 to A.8, we provide additional qualitative results. Figs. A.1 to A.3 show successful navigation to *"stairs"*, *"table"*, and *"television"* (respectively), highlighting the versatility of our ZSON agent.

Figs. A.4 to A.8 illustrate various failure modes of the learned policy. In Fig. A.4, the agent successfully finds a *"refrigerator"* in 4 out of 5 trials. However, in one case the agent stops before entering the kitchen, despite having a view of the *"refrigerator"*. In this case, the failure mode is stopping short.

In Fig. A.5, the agent successfully finds a *"bathtub"* when it enters the bathroom nearest to the starting position. However, in 2 of 5 trials it fails to find that bathroom, thus does not find a *"bathtub."* In this case, the failures are due to poor exploration. Similarly, in Fig. A.6 the agent does find a seating area that resembles a *"sofa"* in 3 of 5 trials. However, it stops in the dining area in 2 trials. Furthermore, it never enters the room to the right, which contains two sofas – again, demonstrating poor exploration.

In Fig. A.7 the agent never enters the room to the bottom right, which contains a *"desk,"* thus failing in all five runs. In 3 of 5 trials it stops near a table that does not appear to be functioning as a desk (e.g., there is no chair nearby). In these examples, the agent might be confusing objects that can have a similar appearance. Finally, in Fig. A.8, we start the agent from two different positions (A and B) and provide the instruction *"Find a dresser."* When the agent is initialized near the bedroom (A) it is able to find the dresser in 4 out of 5 trials.

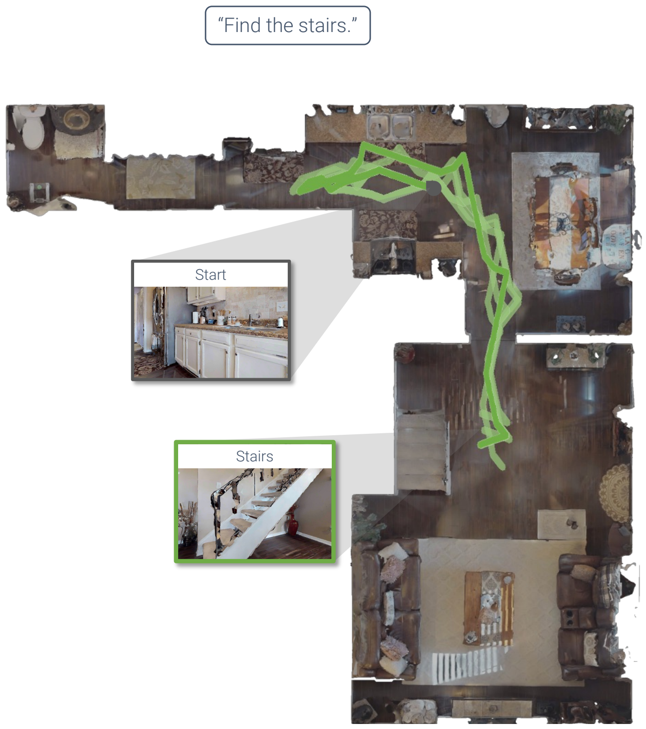

Figure A.1: Qualitative example of successful navigation to the *"stairs."* The number of steps taken by our agent over five trials ranges from 80 to 102.

However, from position B the agent navigates into the kitchen and stops near the cabinets.

Again, these failures may be due to similarities in the appearance of dressers and cabinets.

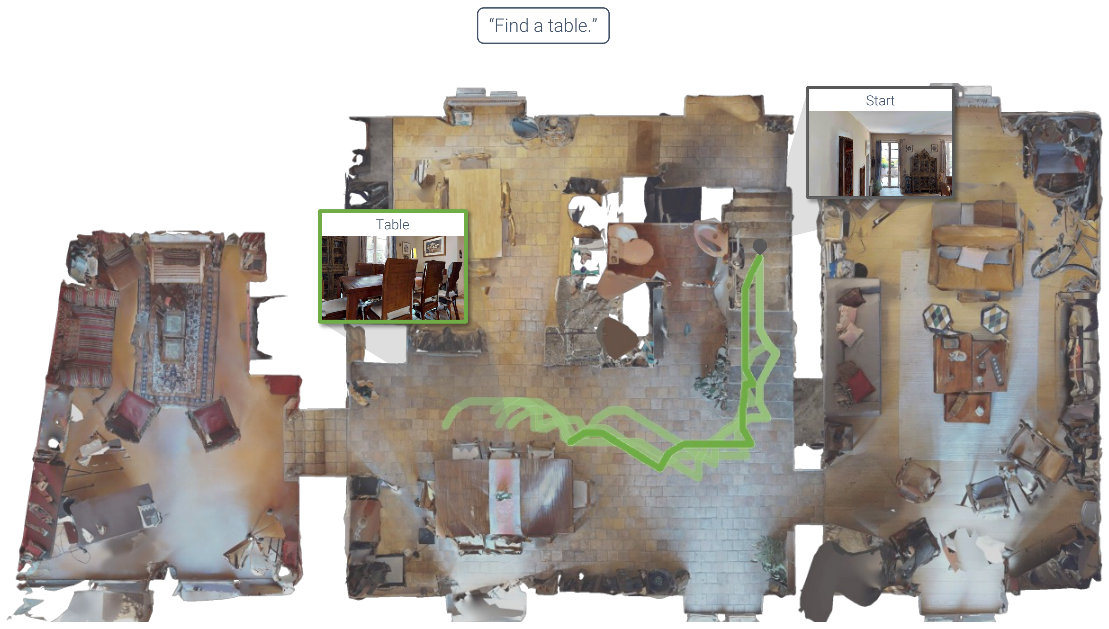

Figure A.2: Qualitative example of successful navigation to a *"table."* The number of steps taken by our agent over five trials ranges from 60 to 98.

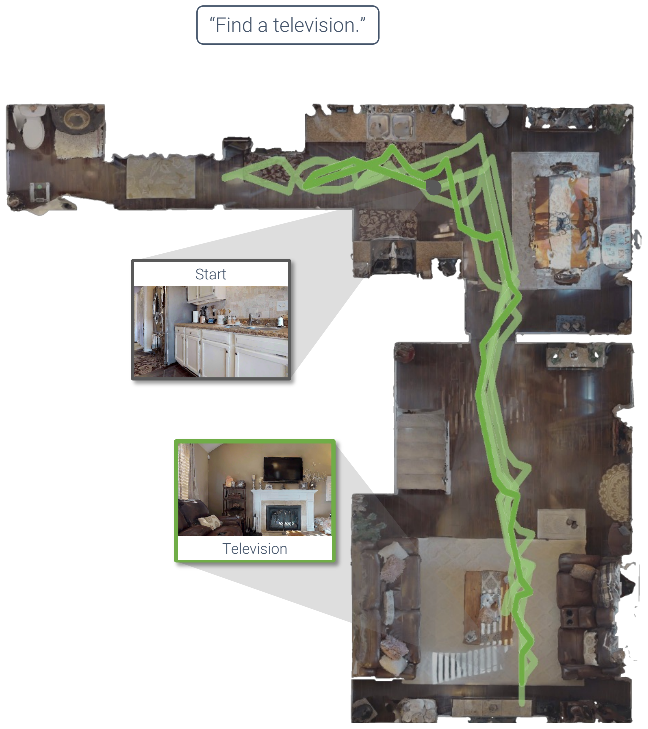

Figure A.3: Qualitative example of successful navigation to a *"television."* The number of steps taken by our agent over five trials ranges from 78 to 148.

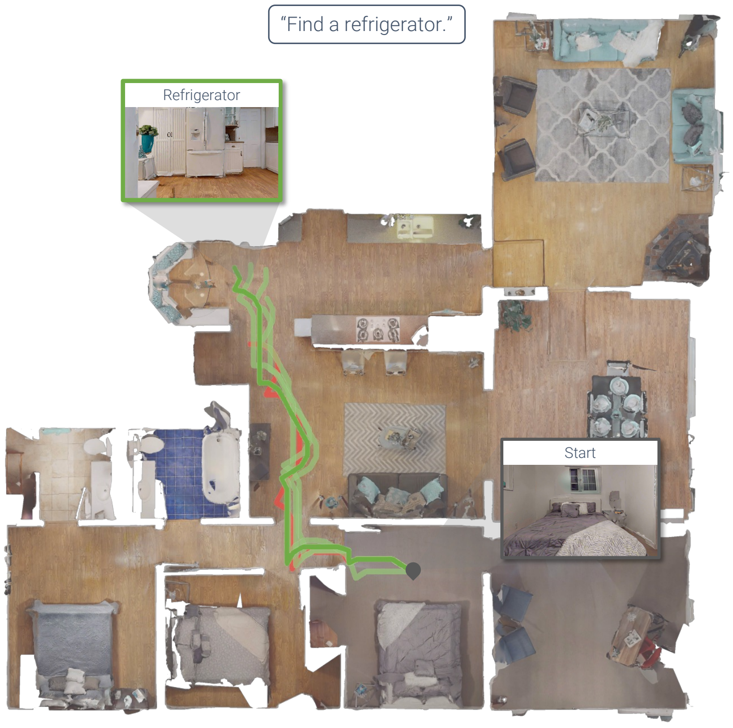

Figure A.4: Qualitative example of *"Find a refrigerator."* The agent succeeds in 4 of 5 trials (green) from the same starting position. In the failure (red) the agent stop short of fridge. The number of steps ranges from 83 to 99.

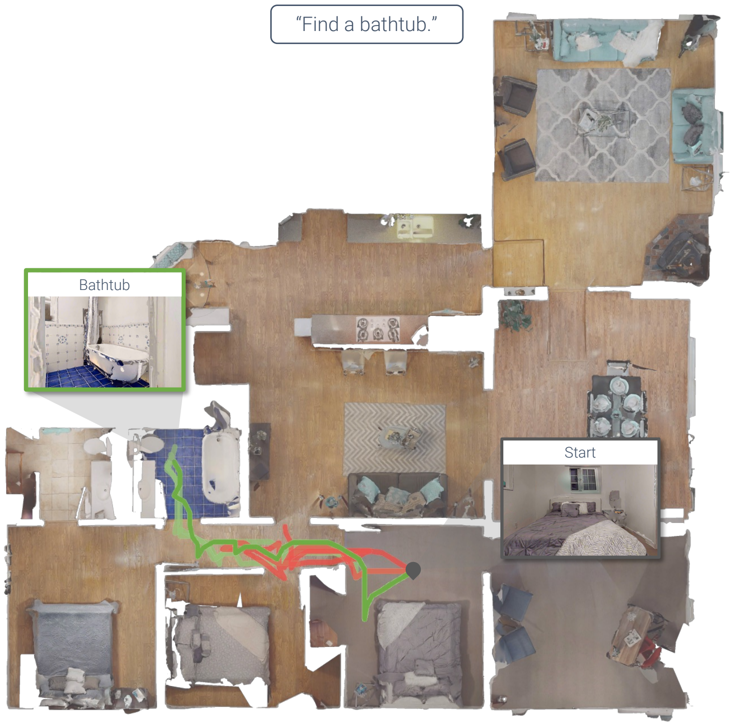

Figure A.5: Qualitative example of *"Find a bathtub."* The agent succeeds in 3 of 5 trials (green) from the same starting position. In the two failures (red) the agent never enters the bathroom. The number of steps ranges from 58 to 114.

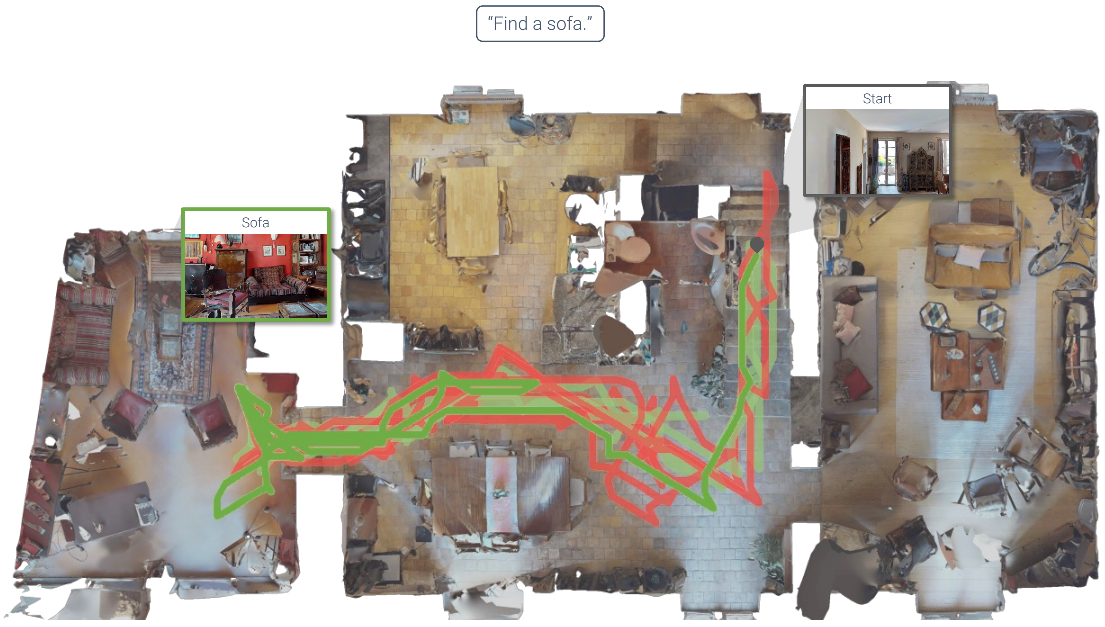

Figure A.6: Qualitative example of *"Find a sofa."* The agent succeeds in 3 of 5 trials (green) from the same starting position. In the two failures (red) the agent stops in the dining area (center). The agent never enters the room to the right with two *"sofas"*. The number of steps ranges from 174 to 501.

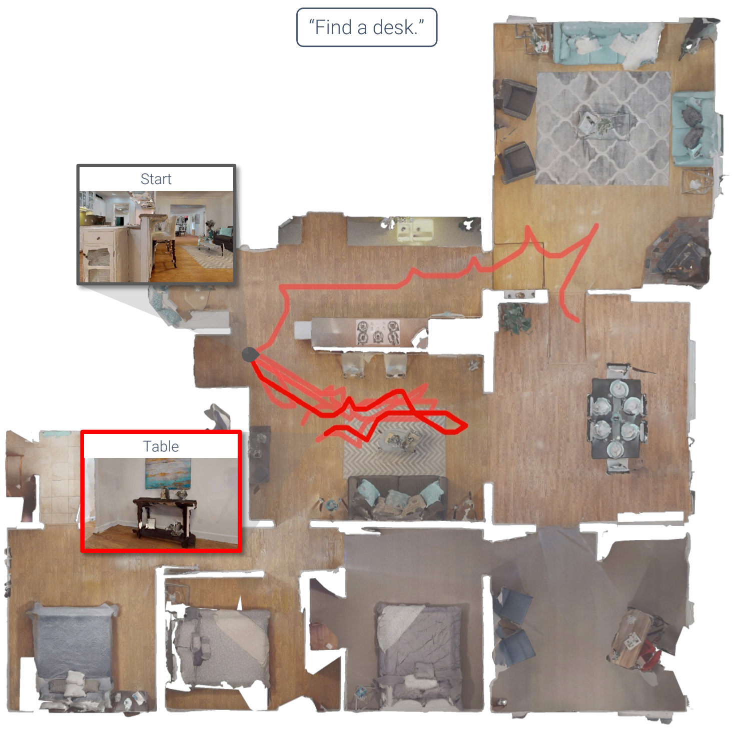

Figure A.7: Qualitative example of *"Find a desk."* The agent fails in all five trials (red), stopping at a table in 3 of 5 runs. The agent never enters the room in the bottom right that contains a *"desk"*. The number of steps ranges from 49 to 114.

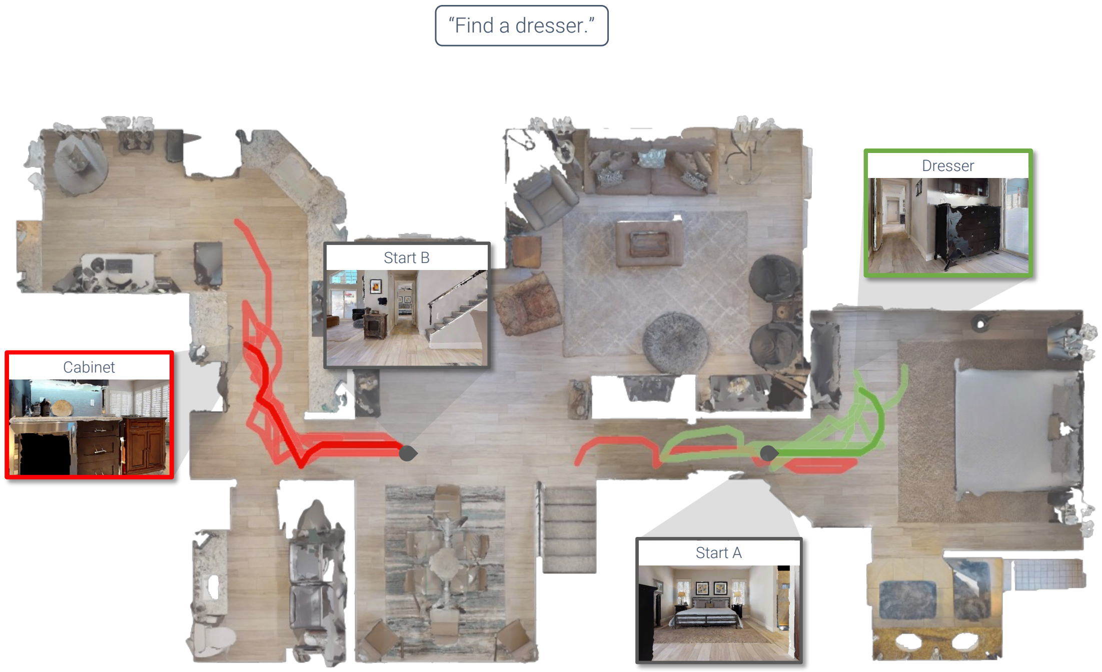

Figure A.8: Qualitative example of *"Find a dresser."* We run five trails from two starting positions (A and B). From Start A the agent is able to find a dresser in 4 of 5 trials (green). However, when the starting location is shifted further from the bedroom (Start B) the agent enters the kitchen and fails in all five runs (red), stopping near the kitchen cabinets. These failures might be due to the similarity in appearance between cabinets and dressers. The number of steps ranges from 29 to 109.

REFERENCES

[1] D. Batra *et al.*, "Objectnav Revisited: On Evaluation of Embodied Agents Navigating to Objects," *arXiv preprint arXiv:2006.13171*, 2020.

[2] M. Savva *et al.*, "Habitat: A platform for embodied ai research," in *ICCV*, 2019.

[3] A. Szot *et al.*, "Habitat 2.0: Training home assistants to rearrange their habitat," *NeurIPS*, 2021.

[4] F. Xia, A. R. Zamir, Z. He, A. Sax, J. Malik, and S. Savarese, "Gibson Env: Real-World Perception for Embodied Agents," in *CVPR*, 2018, pp. 9068–9079.

[5] E. Kolve *et al.*, "AI2-THOR: An Interactive 3D Environment for Visual AI," *arXiv*, 2017.

[6] B. Talbot *et al.*, *BenchBot: Evaluating Robotics Research in Photorealistic 3D Simulation and on Real Robots*, 2020. arXiv: 2008.00635 `[cs.RO]`.

[7] A. X. Chang *et al.*, "ShapeNet: An Information-Rich 3D Model Repository," Stanford University — Princeton University — Toyota Technological Institute at Chicago, Tech. Rep. arXiv:1512.03012 [cs.GR], 2015.

[8] A. Chang *et al.*, "Matterport3D: Learning from RGB-D Data in Indoor Environments," in *ThreeDV*, MatterPort3D dataset license: http://kaldir.vc.in.tum.de/matterport/MP_TOS.pdf, 2017.

[9] I. Armeni *et al.*, "3D Scene Graph: A Structure for Unified Semantics, 3D Space, and Camera," in *ICCV*, 2019.

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

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

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

[13] O. Maksymets *et al.*, "Thda: Treasure hunt data augmentation for semantic navigation," in *ICCV*, 2021.

[14] Y. Liang, B. Chen, and S. Song, "SSCNav: Confidence-Aware Semantic Scene Completion for Visual Semantic Navigation," in *ICRA*, 2021.

[15] H. Luo, A. Yue, Z.-W. Hong, and P. Agrawal, “Stubborn: A Strong Baseline for Indoor Object Navigation,” *arXiv preprint arXiv:2203.07359*, 2022.

[16] K. Yadav *et al.*, “Offline Visual Representation Learning for Embodied Navigation,” *arXiv preprint arXiv:2204.13226*, 2022.

[17] K. Yadav *et al.*, *Habitat challenge 2022*, https://aihabitat.org/challenge/2022/, 2022.

[18] A. Radford *et al.*, “Learning Transferable Visual Models from Natural Language Supervision,” in *ICML*, 2021.

[19] S. K. Ramakrishnan *et al.*, “Habitat-matterport 3d dataset (HM3d): 1000 large-scale 3d environments for embodied AI,” in *NeurIPS Datasets and Benchmarks Track*, 2021.

[20] Z. Al-Halah, S. K. Ramakrishnan, and K. Grauman, “Zero Experience Required: Plug & Play Modular Transfer Learning for Semantic Visual Navigation,” *arXiv preprint arXiv:2202.02440*, 2022.

[21] S. Y. Gadre, M. Wortsman, G. Ilharco, L. Schmidt, and S. Song, “CLIP on Wheels: Zero-Shot Object Navigation as Object Localization and Exploration,” *arXiv preprint arXiv:2203.10421*, 2022.

[22] C. Jia *et al.*, “Scaling Up Visual and Vision-Language Representation Learning With Noisy Text Supervision,” in *ICML*, 2021.

[23] H. Pham *et al.*, “Combined Scaling for Zero-shot Transfer Learning,” *arXiv preprint arXiv:2111.10050*, 2021.

[24] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, “Imagenet: A Large-Scale Hierarchical Image Database,” in *CVPR*, 2009.

[25] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi, “Simple but Effective: CLIP Embeddings for Embodied AI,” *arXiv preprint arXiv:2111.09888*, 2021.

[26] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh, and D. Batra, “Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization,” in *ICCV*, 2017.

[27] A. v. d. Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding,” *arXiv preprint arXiv:1807.03748*, 2018.

[28] Y. Zhu *et al.*, “Target-driven Visual Navigation in Indoor Scenes using Deep Reinforcement Learning,” *ICRA*, 2017.

[29] K. He, X. Zhang, S. Ren, and J. Sun, “Deep Residual Learning for Image Recognition,” in *CVPR*, 2016.

[30] A. Eftekhar, A. Sax, J. Malik, and A. Zamir, “Omnidata: A Scalable Pipeline for Making Multi-Task Mid-Level Vision Datasets From 3D Scans,” in *ICCV*, 2021.

[31] M. Caron *et al.*, “Emerging Properties in Self-Supervised Vision Transformers,” in *ICCV*, 2021.

[32] E. Wijmans *et al.*, “DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames,” in *ICLR*, 2019.

[33] L. Mezghani *et al.*, “Memory-Augmented Reinforcement Learning for Image-Goal Navigation,” *arXiv preprint arXiv:2101.05181*, 2021.

[34] P. Anderson *et al.*, “On Evaluation of Embodied Navigation Agents,” *arXiv preprint arXiv:1807.06757*, 2018.

[35] A. Paszke *et al.*, “Pytorch: An imperative style, high-performance deep learning library,” in *NeurIPS*, 2019.

[36] B. Yamauchi, “A frontier-based approach for autonomous exploration,” in *CIRA*, 1997.

[37] R. Ramrakhya, E. Wijmans, D. Batra, and A. Das, *Not all demonstrations are created equal: An objectnav case study for effectively combining imitation and reinforcement learning*, https://github.com/Ram81/il_rl_baselines, 2022.