LM-Nav: Robotic Navigation with Large Pre-Trained Models of Language, Vision, and Action

Dhruv Shah[†β], **Błażej Osiński**[†βω], **Brian Ichter**[γ], **Sergey Levine**[βγ]
[β]UC Berkeley, [ω]University of Warsaw, [γ]Robotics at Google

Abstract: Goal-conditioned policies for robotic navigation can be trained on large, unannotated datasets, providing for good generalization to real-world settings. However, particularly in vision-based settings where specifying goals requires an image, this makes for an unnatural interface. Language provides a more convenient modality for communication with robots, but contemporary methods typically require expensive supervision, in the form of trajectories annotated with language descriptions. We present a system, LM-Nav, for robotic navigation that enjoys the benefits of training on unannotated large datasets of trajectories, while still providing a high-level interface to the user. Instead of utilizing a labeled instruction following dataset, we show that such a system can be constructed entirely out of pre-trained models for navigation (ViNG), image-language association (CLIP), and language modeling (GPT-3), without requiring any fine-tuning or language-annotated robot data. LM-Nav extracts landmarks names from an instruction, grounds them in the world via the image-language model, and then reaches them via the (vision-only) navigation model. We instantiate LM-Nav on a real-world mobile robot and demonstrate long-horizon navigation through complex, outdoor environments from natural language instructions.

1 Introduction

One of the central challenges in robotic learning is to enable robots to perform a wide variety of tasks on command, following high-level instructions from humans. This requires robots that can understand human instructions, and are equipped with a large repertoire of diverse behaviors to execute such instructions in the real world. Prior work on instruction following in navigation has largely focused on learning from trajectories annotated with textual instructions [1–5]. This enables understanding of textual instructions, but the cost of data annotation impedes wide adoption. On the other hand, recent work has shown that learning robust navigation is possible through goal-conditioned policies trained with self-supervision. These utilize large, unlabeled datasets to train vision-based controllers via hindsight relabeling [6–11]. They provide scalability, generalization, and robustness, but usually involve a clunky mechanism for goal specification, using locations or images. In this work, we aim to combine the strengths of both approaches, enabling a robotic navigation system to execute natural language instructions by leveraging the capabilities of pre-trained models *without any user-annotated navigational data*. Our method uses these models to construct an "interface" that humans can use to communicate desired tasks to robots. This system enjoys the impressive generalization capabilities of the pre-trained language and vision-language models, enabling the robotic system to accept complex high-level instructions.

Our main observation is that we can utilize off-the-shelf *pre-trained models* trained on large corpora of visual and language datasets — that are widely available and show great few-shot generalization capabilities — to create this interface for embodied instruction following. To achieve this, we combine the strengths of two such robot-agnostic pre-trained models with a pre-trained navigation model. We use a visual navigation model (**VNM**: ViNG [11]) to create a topological "mental map" of the environment using the robot's observations from a prior exploration of the environment. Given free-form textual instructions, we use a pre-trained large language model (**LLM**: GPT-3 [12]) to decode the instructions into a sequence of textual landmarks. We then use a vision-language model

[†] These authors contributed equally, order decided by a coin flip. Check out the project page for experiment videos, code, and a user-friendly Colab notebook that runs in your browser: sites.google.com/view/lmnav

6th Conference on Robot Learning (CoRL 2022), Auckland, New Zealand.

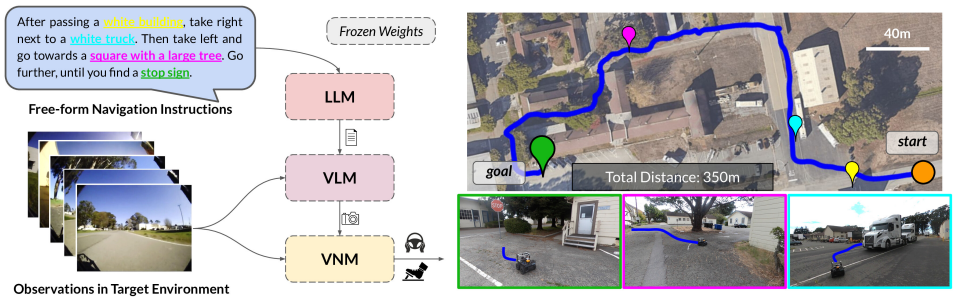

Figure 1: Embodied instruction following with LM-Nav: Our system takes as input a set of raw observations from the target environment and free-form textual instructions (left), deriving an actionable plan using three *pre-trained* models: a large language model (**LLM**) for extracting landmarks, a vision-and-language model (**VLM**) for grounding, and a visual navigation model (**VNM**) for execution. This enables LM-Nav to follow textual instructions in complex environments purely from visual observations (right) *without any fine-tuning*.

(**VLM**: CLIP [13]) for *grounding* these textual landmarks in the topological map, by inferring a joint likelihood over the landmarks and nodes. A novel search algorithm is then used to plan a path for the robot, which is then executed by **VNM**. While reducing the task of language following to a combination of grounding and subgoal selection discards a lot of useful cues such as relations and verbs, we find that it is still sufficient to follow a variety of natural language instructions.

Our primary contribution is **L**arge **M**odel **Nav**igation, or LM-Nav, an embodied instruction following system that combines three large independently pre-trained models — a robotic control model that utilizes visual observations and physical actions (**VNM**), a vision-language model that grounds images in text but has no context of embodiment (**VLM**), and a large language model that can parse and translate text but has no sense of visual grounding or embodiment (**LLM**) — to enable long-horizon instruction following in complex, real-world environments. *We present the first instantiation of a robotic system that combines the confluence of pre-trained vision-and-language models with a goal-conditioned controller, to derive actionable plans* without any fine-tuning *in the target environment.* Notably, all three models are trained on large-scale datasets, with self-supervised objectives, and used off-the-shelf with *no fine-tuning* — no human annotations of the robot navigation data are necessary to train LM-Nav. We show that LM-Nav is able to successfully follow natural language instructions in pre-explored environments over the course of 100s of meters of complex, suburban navigation, while disambiguating paths with fine-grained commands.

2 Related Work

Early works in augmenting navigation policies with natural language commands use statistical machine translation [14] to discover data-driven patterns to map free-form commands to a formal language defined by a grammar [15–19]. However, these approaches tend to operate on structured state spaces. Our work is closely inspired by methods that instead reduce this task to a sequence prediction problem [1, 20, 21]. Notably, our goal is similar to the task of VLN — leveraging fine-grained instructions to control a mobile robot solely from visual observations [1, 2].

However, most recent approaches to VLN use a large dataset of simulated trajectories — over 1M demonstrations — annotated with fine-grained language labels in indoor [1, 3–5, 22] and driving scenarios [23–28], and rely on sim-to-real transfer for deployment in simple indoor environments [29, 30]. However, this necessitates building a photo-realistic simulator resembling the target environment, which can be challenging for unstructured environments, especially for the task of outdoor navigation. Instead, LM-Nav leverages free-form textual instructions to navigate a robot in complex, outdoor environments *without* access to any simulation or any trajectory-level annotations.

Recent progress in using large-scale models of natural language and images trained on diverse data has enabled applications in a wide variety of textual [31–33], visual [13, 34–38], and embodied domains [39–44]. In the latter category, approaches either fine-tune embeddings from pre-trained models on robot data with language labels [39, 40, 44], assume that the low-level agent can execute textual instructions (without addressing control) [41], or assume access to a set of text-conditioned skills that can follow atomic textual commands [42]. All of these approaches require access to low-level skills that can follow rudimentary textual commands, necessitating language annotations for

robotic experience and a strong assumption on the robot's capabilities. In contrast, we combine these pre-trained vision and language models with pre-trained visual policies that do not use any language annotations [11, 45] *without* fine-tuning these models for the task of VLN.

Data-driven approaches to vision-based mobile robot navigation often use photorealistic simulators [46–49] or supervised data collection [50] to learn goal-reaching policies directly from raw observations. Self-supervised methods for navigation [6–11, 51] instead can use unlabeled datasets of trajectories by automatically generating labels using onboard sensors and hindsight relabeling. While such policies are adept at navigating to goal locations or images, they may be unable to parse high-level instructions such as free-form text. LM-Nav uses self-supervised policies trained in a large number of prior environments, augmented with pre-trained vision and language models for parsing natural language instructions, and deploys them in novel real-world environments *without* any fine-tuning. We emphasize that while LM-Nav relies on a pre-built topological graph, similar to prior work [11, 51, 52], this assumption may be relaxed by incorporating exploration heuristics in unseen environments [53], and can be an interesting avenue for future work.

3 Preliminaries

LM-Nav consists of three large, pre-trained models for processing language, associating images with language, and visual navigation.

Large language models are generative models of text trained on large corpora of internet text using self-supervised learning. LM-Nav uses the GPT-3 **LLM** [12] to parse instructions into a sequence of landmarks.

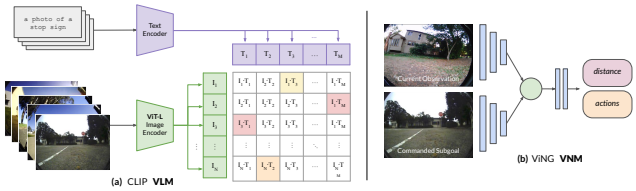

Figure 2: LM-Nav uses CLIP to infer a joint distribution over textual landmarks and image observations. **VNM** infers a goal-conditioned distance function and policy that can control the robot.

Vision-and-language models refer to models that can associate images and text, e.g. image captioning, visual question-answering, etc. [54–56]. We use the CLIP **VLM** [13], a model that jointly encodes images and text into a shared embedding space, to jointly encode a set of landmark descriptions t obtained from the **LLM** and a set of images i_k to obtain their **VLM** embeddings $\{T, I_k\}$ (see Fig. 3). Computing the cosine similarity between these embeddings, followed by a softmax operation results in probabilities $P(i_k|t)$, corresponding to the likelihood that image i_k corresponds to the string t. LM-Nav uses this probability to align landmark descriptions with images.

Visual navigation models learn navigational affordances directly from visual observations [11, 51, 57–59], associating images and actions through time. We use the ViNG **VNM** [11], a goal-conditioned model that predicts temporal distances between pairs of images and the corresponding actions to execute (see Fig. 3). The **VNM** serves two purposes: (i) given a set of observations in the target environment, the distance predictions from the **VNM** can be used to construct a topological graph $\mathcal{G}(V, E)$ that represents a "mental map" of the environment; (ii) given a "walk" (i.e., a sequence of connected subgoals to the goal), **VNM** can control the robot along this plan. The topological graph $\mathcal{G}$ is an important abstraction that allows a simple interface for planning over past experience in the environment and has been successfully used in prior work to perform long-horizon navigation [52, 53, 60]. To deduce connectivity in $\mathcal{G}$, we use a combination of learned distance estimates, temporal proximity (during data collection), and spatial proximity (using GPS measurements). For more details on the construction of this graph, see Appendix B.

4 LM-Nav: Instruction Following with Pre-Trained Models

LM-Nav combines the components discussed earlier to follow natural language instructions in the real world. The **LLM** parses free-form instructions into a list of landmarks $\bar{\ell}$ (Sec. 4.2), the **VLM** associates these landmarks with nodes in the graph by estimating the probability that each node $\bar{v}$ corresponds to each $\bar{\ell}$, $P(\bar{v}|\bar{\ell})$ (Sec. 4.3), and the **VNM** is used to infer how effectively the robot can navigate between each pair of nodes in the graph, denoted by a probability $P(\overrightarrow{v_i, v_j})$. To find the optimal "walk" on the graph that both (i) adheres to the provided instructions and (ii) minimizes traversal cost, we derive a probabilistic objective (Sec. 4.1) and show how it can be optimized using a graph search algorithm (Sec. 4.4). This walk is executed in the real world by the **VNM** model.

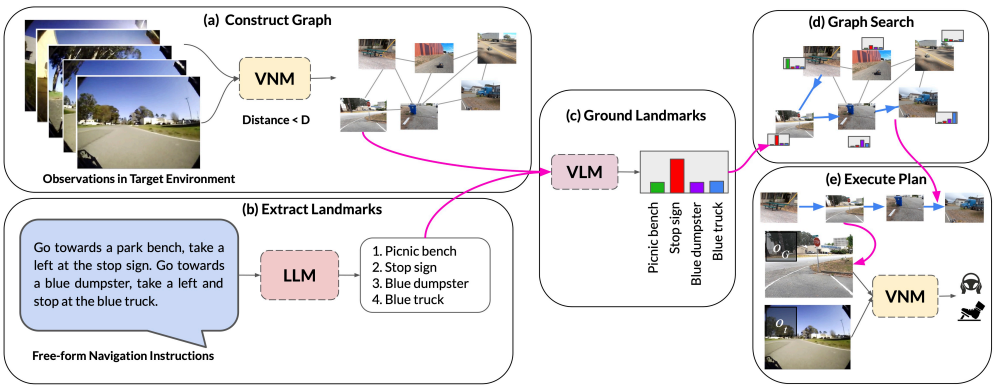

Figure 3: System overview: (a) **VNM** uses a goal-conditioned distance function to infer connectivity between the set of raw observations and constructs a topological graph. (b) **LLM** translates natural language instructions into a sequence of textual landmarks. (c) **VLM** infers a joint probability distribution over the landmark descriptions and nodes in the graph, which is used by (d) a graph search algorithm to derive the optimal walk through the graph. (e) The robot drives following the walk in the real world using the **VNM** policy.

4.1 Problem Formulation

Given a sequence of landmark descriptions $\bar{\ell} = \ell_1, \ell_2, ..., \ell_n$ extracted from the language command, our method needs to determine a sequence of waypoints $\bar{v} = v_1, v_2, ..., v_k$ to command to the robot. Typically, $k \geq n$, since each landmark needs to be visited, but the traversal might require other waypoints in between the landmarks. Finding $\bar{v}$ can formulated as a probabilistic inference problem. A key element in this formulation is access to a distribution $p(v_i|\ell_j)$ for each graph vertex v_i and landmark description ℓ_j. Recall that the graph vertices correspond to images observed by the robot, and thus, $p(v_i|\ell_i)$ represents a distribution over images given a language description. This can be obtained from the **VLM**. Intuitively, the full likelihood that we need to optimize to determine the robot's plan will now depend on two terms: likelihoods of the form $p(v_{t_i}|\ell_i)$ that describe how likely v_{t_i} is to correspond to ℓ_i for an assignment $t_1, t_2, \ldots, t_n$, and traversability likelihoods $p(\overline{v_i, v_{i+1}})$ that describe how likely is the robot to be able to reach v_{i+1} from v_i.

While we can use a variety of traversability likelihood functions, a simple choice is to use a discounted Markovian model, where the discount γ models the probability of *exiting* at each time step, leading to a termination probability of $1 - \gamma$ at each step, and a probability of reaching v_{i+1} given by $\gamma^{D(v_i, v_{i+1})}$, where $D(v_i, v_{i+1})$ is the estimated number of time steps the robot needs to travel from v_i to v_{i+1}, which is predicted by the **VNM**. While other traversability likelihoods could also be used, this choice is a convenient consequence of goal-conditioned reinforcement learning formulations [61, 62], and thus, the log-likelihood corresponds to $D(v_i, v_{i+1})$. We can use these likelihoods to derive the probability that a given sequence $\bar{v}$ can be traversed successfully, which we denote with the auxiliary Bernoulli random variable $c_{\bar{v}}$ (i.e., $c_{\bar{v}} = 1$ implies that $\bar{v}$ was traversed successfully):

$$P(c_{\bar{v}} = 1|\bar{v}) = \prod_{1 \leq i < T} P(\overline{v_i, v_{i+1}}) = \prod_{1 \leq i < T} \gamma^{D(v_i, v_{i+1})}, \tag{1}$$

The full likelihood used for planning is then given by:

$$P(\text{success}|\bar{v}, \bar{\ell}) \propto P(c_{\bar{v}} = 1|\bar{v}) P(\bar{v}|\bar{\ell}) = \prod_{1 \leq j < k} \gamma^{D(v_j, v_{j+1})} \max_{1 \leq t_1 \leq \ldots \leq t_n \leq k} \prod_{1 \leq i \leq n} P(v_{t_i}|\ell_i). \tag{2}$$

4.2 Parsing Free-Form Textual Instructions

The user specifies the route they want the robot to take using natural language, while the objective above is defined in terms of a sequence of desired landmarks. To extract this sequence from the user's natural language instruction we employ a large language model, which in our prototype is GPT-3 [12]. We used a prompt with 2 examples of correct landmarks' extractions, followed by the description to be translated by the **LLM**. Examples of instructions and landmarks extracted by the model can be found in Fig. 4. The prompt was selected to disambiguate nuanced cases, e.g. when order of landmarks in the text is different than in the expected path (see example in Fig. 4 a). For details of the *"prompt engineering"* please see Appendix A.

4.3 Visually Grounding Landmark Descriptions

As discussed in Sec. 4.1, a crucial element of selecting the walk through the graph is computing $P(v_i|\ell_j)$, the probability that landmark description v_i refers to node ℓ_j (see Eqn. 2). With each node containing an image taken during initial data collection, the probability can be computed using CLIP [13] in the way described in Sec. 3 as the retrieval task. As presented in Fig. 2, we apply CLIP to the image at node v_i and caption prompt in the form of *"This is a photo of a [ℓ_j]"*. To go from CLIP model outputs, which are logits, to probabilities we use $P(v_i|\ell_j) = \frac{\exp \text{CLIP}(v_i,\ell_j)}{\sum_{v\in V} \exp \text{CLIP}(v,\ell_j)}$. The resulting probability $P(v_i|\ell_j)$, together with the inferred edges' distances will be used to select the optimal walk.

Algorithm 1: Graph Search

Input: Landmarks $(\ell_1, \ell_2, \ldots, \ell_n)$.
Input: Graph $\mathcal{G}(V, E)$.
Input: Starting node S.
$\forall_{\substack{i=0,\ldots,n \\ v\in V}} \; Q[i, v] = -\infty$
$Q[0, S] = 0$
Dijkstra_algorithm($\mathcal{G}, Q[0, *]$)
for i in $1, 2, \ldots, n$ **do**
  $\forall v \in V Q[i, v] = Q[i - 1, v] + \text{CLIP}(v, \ell_i)$
  Dijkstra_algorithm($\mathcal{G}, Q[i, *]$)
end for
destination $= \arg\max(Q[n, *])$
return backtrack(destination, $Q[n, *]$)

4.4 Graph Search for the Optimal Walk

As described in Sec. 4.1, LM-Nav aims at finding a walk $\bar{v} = (v_1, v_2, \ldots, v_k)$ that maximizes the probability of successful execution of $\bar{v}$ that adheres to the given list of landmarks $\bar{\ell}$. We can define a function $R(\bar{v}, \bar{t})$ for a monotonically increasing sequence of indices $\bar{t} = (t_1, t_2, \ldots, t_n)$:

$$R(\bar{v}, \bar{t}) := \sum_{i=1}^{n} \text{CLIP}(v_{t_i}, \ell_i) - \alpha \sum_{j=1}^{T-1} D(v_j, v_{j+1}), \text{where } \alpha = -\log \gamma. \tag{3}$$

R has the property that $(\bar{v})$ maximizes $P(\text{success}|\bar{v}, \bar{\ell})$ defined in Eqn. 2, if and only if there exists $\bar{t}$ such that $(\bar{v}, \bar{t})$ maximizes R. In order to find such $(\bar{v}, \bar{t})$, we employ dynamic programming. In particular we define a helper function $Q(i, v)$ for $i \in \{0, 1, \ldots, n\}$, $v \in V$:

$$Q(i, v) = \max_{\substack{\bar{v}=(v_1,v_2,\ldots,v_j),v_j=v \\ \bar{t}=(t_1,t_2,\ldots,t_i)}} R(\bar{v}, \bar{t}). \tag{4}$$

$Q(i, v)$ represents the maximal value of R for a walk ending in v that visited the landmarks up to index i. The base case $Q(0, v)$ visits none of the landmarks, and its value of R is simply equal to minus the length of shortest path from the starting node S. For $i > 0$ we have:

$$Q(i, v) = \max\left(Q(i - 1, v) + \text{CLIP}(v, \ell_i), \max_{w\in\text{neighbors}(v)} Q(i, w) - \alpha \cdot D(v, w)\right). \tag{5}$$

The base case for DP is to compute $Q(0, V)$. Then, in each step of DP $i = 1, 2, \ldots, n$ we compute $Q(i, v)$. This computation resembles the Dijkstra algorithm ([63]). In each iteration, we pick the node v with the largest value of $Q(i, v)$ and update its neighbors based on the Eqn. 5. Algorithm 1 summarizes this search process. The result of this algorithm is a walk $\bar{v} = (v_1, v_2, \ldots, v_k)$ that maximizes the probability of successfully carrying out the instruction. Such a walk can be executed by **VNM**, using its action estimates to sequentially navigate to these nodes.

5 System Evaluation

We now describe our experiments deploying LM-Nav in a variety of outdoor settings to follow high-level natural language instructions with a small ground robot (Clearpath Jackal UGV platform — see Fig. 1(right) for image and Appendix C for details). For all experiments, the weights of **LLM**, **VLM**, and **VNM** are frozen — there is *no fine-tuning or annotation* in the target environment. We evaluate the complete system, as well as the individual components of LM-Nav, to understand its strengths and limitations. Our experiments demonstrate the ability of LM-Nav to follow high-level instructions, disambiguate paths, and reach goals that are up to 800m away.

Figure 4: Qualitative examples of LM-Nav in real-world environments executing textual instructions (left). The landmarks extracted by **LLM** (highlighted in text) are grounded into visual observations by **VLM** (center; overhead image not available to the robot). The resulting *walk* of the graph is executed by **VNM** (right).

5.1 Following Instructions with LM-Nav

In each evaluation environment, we first construct the graph by manually driving the robot and collecting image and GPS observations. The graph is constructed automatically using the **VNM** to predict relative distances between images in these trajectories. We tested our system on 20 queries in 2 environments, corresponding to a combined length of over 6km. The instructions include prominent landmarks that can be identified from the robot's observations, e.g., buildings and stop signs.

Fig. 4 shows qualitative examples of the path taken by the robot. In Fig. 4(a), LM-Nav is able to successfully localize the simple landmarks from its prior traversal and find a short path to the goal. While there are multiple stop signs in the environment, the objective in Eqn. 2 causes the robot to pick the correct one, minimizing overall trajectory length. Fig. 4(b) highlights LM-Nav's ability to follow complex instructions with multiple landmarks — despite the possibility of taking a shorter route directly to the final landmark, the robot follows a path that correctly visits all of the landmarks.

Missing landmarks. While LM-Nav is effective at finding a path through landmarks extracted from instructions, it relies on the assumption that the landmarks (i) exist in the environment, and (ii) can be identified by the **VLM**. Fig. 4(c) illustrates a case where the executed path fails to visit one of the landmarks — a fire hydrant — and takes a path that goes around the top of the building rather than the bottom. This failure mode is attributed to the the inability of the **VLM** to detect a fire hydrant from the robot's observations. On independently evaluating the efficacy of the **VLM** at retrieving landmarks (see Sec. 5.3), we find that despite being the best off-the-shelf model for our task, CLIP is unable to retrieve a small number of "hard" landmarks, including fire hydrants and cement mixers. In many practical cases, the robot is still successful in finding a path that visits the remaining landmarks.

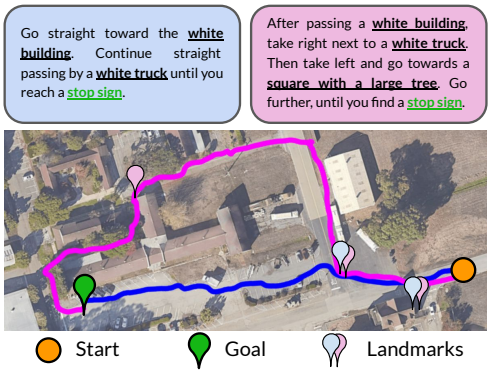

Figure 5: LM-Nav can successfully disambiguate instructions with same start-goal locations that differ slightly. The landmarks are underscored in text and their locations are marked with pins.

Disambiguation with instructions. Since the objective of LM-Nav is to follow instructions, and not merely to reach the final goal, different instructions may lead to different traversals. Fig. 5 shows an example where modifying the instruction can disambiguate multiple paths to the goal. Given the shorter prompt (blue), LM-Nav prefers the more direct path. On specifying a more fine-grained route (magenta), LM-Nav takes an alternate path that passes a different set of landmarks.

5.2 Quantitative Analysis

To quantify the performance of LM-Nav, we introduce the following metrics. A walk found by the graph search is *successful*, if (1) it matches the path intended by the user or (2) if the landmark images extracted by the search algorithm contain said landmarks (i.e. if the path visits landmarks with

System	Environment	Net Success ↑	Efficiency ↑	# Diseng. ↓	Planning ↑
GPS-Nav (No **VNM**)	`EnvSmall-10`	0.23	0.93	0.75	**0.9**
LM-Nav (Ours)	`EnvSmall-10`	**0.8**	**0.96**	**0.1**	**0.9**
	`EnvLarge-10`	**0.8**	**0.89**	**0**	**0.8**

Table 1: Quantifying navigational instruction following with LM-Nav over 20 experiments. LM-Nav can successfully plan a path to the goal, and follow it efficiently, over 100s of meters. Ablating the **VNM** (GPS-Nav) severely hurts performance due to frequent disengagements inability to reason about collisions with obstacles.

LLM Candidate	Avg. Extraction Success
Noun Chunks	0.88
fairseq-1.3B [64]	0.52
fairseq-13B [64]	0.76
GPT-J-6B [65]	0.80
GPT-NeoX-20B [66]	0.72
GPT-3 [12]	**1.0**

Table 2: GPT-3 consistently outperforms alternatives in parsing free-form instructions into landmarks.

VLM Candidate	Detection Rate
Faster-RCNN [67]	0.07
ViLD [36]	0.38
CLIP-ViT [13]	**0.87**

Table 3: CLIP-ViT produces the most reliable landmark detections from visual observations.

the same description, even if not exactly the same). *Planning success* is the fraction of successful walks found by the search algorithm. *Efficiency* of a walk is defined as the ratio of the lengths of the described route and the executed one; the value is clipped at a maximum of 1 to account for the cases when the LM-Nav executes a path shorter than the user intended. For a set of queries, we report the average efficiency over successful experiments. The *planning efficiency* is similarly defined as the ratio of the length of the described and *planned* routes. Finally, *number of disengagements* is the average number of human interventions per experiment due to unsafe maneuvers.

Table 1 summarizes the quantitative performance of the system over 20 instructions. LM-Nav generates a successful walk for 85% of them, and causes disengagement only once (an average of 1 intervention per 6.4km of traversals). Investigating the planning failure modes suggests that the most critical component of our system is the ability of **VLM** to detect certain landmarks, e.g. a fire hydrant, and in challenging lighting conditions, e.g. underexposed images.

5.3 Dissecting LM-Nav

To understand the influence of each of the components of LM-Nav, we conduct experiments to evaluate these components in isolation. For more details about these experiments, see Appendix D.

To evaluate the performance of **LLM** candidates in parsing instructions into an *ordered* list of landmarks, we compare GPT-3 (used by LM-Nav) to other state-of-the-art pre-trained language models — fairseq [64], GPT-J-6B [65], and GPT-NeoX-20B [66] — as well as a simple baseline using spaCy NLP library [68] that extracts base noun phrases, followed by filtering. In Table 2 we report the average extraction success for all the methods on the 20 prompts used in Section 5.2. GPT-3 significantly outperforms other models, owing to its superior representation capabilities and in-context learning [69]. The noun chunking performs surprisingly reliably, correctly solving many simple prompts. For further details on these experiments, see Appendix D.2.

To evaluate the **VLM**'s ability to ground these textual landmarks in visual observations, we set up an object detection experiment. Given an unlabeled image from the robot's on-board camera and a set of textual landmarks, the task is to *retrieve* the corresponding label. We run this experiment on a set of 100 images from the environments discussed earlier, and a set of 30 commonly-occurring landmarks. These landmarks are a combination of the landmarks retrieved by the **LLM** in our

	`EnvSmall-10`		`EnvLarge-10`	
Planner	Pl. Success ↑	Pl. Efficiency ↑	Pl. Success ↑	Pl. Efficiency ↑
Max Likelihood	0.6	0.69	0.2	0.17
LM-Nav (Ours)	**0.9**	**0.80**	**0.8**	**0.99**

Table 4: Ablating the search algorithm (Sec. 4.4) gives a max likelihood planner that ignores reachability information, resulting in inefficient plans that are up to 6× longer than LM-Nav for the same instruction.

experiments from Sec. 5.1 and manually curated ones. We report the detection successful if any of the top 3 predictions adhere to the contents of the image. We compare the retrieval success of our **VLM** (CLIP) with some object detection alternatives — Faster-RCNN-FPN [67, 70], a state-of-the-art object detection model pre-trained on MS-COCO [71, 72], and ViLD [36], an open-vocabulary object detector based on CLIP and Mask-RCNN [73]. To evaluate against the closed-vocabulary baseline, we modify the setup by projecting the landmarks onto the set of MS-COCO class labels. We find that CLIP outperforms baselines by a wide margin, suggesting that its visual model transfers very well to robot observations (see Table 3). Despite deriving from CLIP, ViLD struggles with detecting complex landmarks like "manhole cover" and "glass building". Faster-RCNN is unable to detect common MS-COCO objects like "traffic light", "person" and "stop sign", likely due to the on-board images being out-of-distribution for the model.

To understand the importance of the **VNM**, we run an ablation experiment of LM-Nav without the navigation model. Using GPS-based distance estimates and a naïve straight line controller between nodes of the topological graph. Table 1 summarizes these results — without **VNM**'s ability to reason about obstacles and traversability, the system frequently runs into small obstacles such as trees and curbs, resulting in failure. Fig. 6 illustrates such a case — while such a controller works well on open roads, it fails to reason about connectivity around buildings or obstacles and results in collisions with a curb, a tree, and a wall in 3 individual attempts. This illustrates that using a learned policy and distance function from the **VNM** is critical for LM-Nav to successfully navigate in complex environments.

Figure 6: GPS-Nav (red) fails to execute a plan due to its inability to reason about traversability through obstacles, while LM-Nav (blue) succeeds.

Lastly, to understand the importance of the two components of the graph search objective (Eqn. 3), we ran a set of ablations where the graph search only depends on $P(\bar{v}|\ell)$, i.e. *Max Likelihood Planning*, which only picks the most likely landmark without reasoning about topological connectivity or traversability. Table 4 shows that such a planner suffers greatly in the form of efficiency, because it does not utilize the spatial organization of nodes and their connectivity. For more details on these experiments, and qualitative examples, see Appendix D.

6 Discussion

We presented **L**arge **M**odel **Nav**igation, a robotic system that can execute textual instructions in the real-world without requiring any human annotations for navigation trajectories. LM-Nav combines three pre-trained models: the **LLM**, which parses instructions into a list of landmarks; the **VLM**, which infers joint probabilities between these landmarks and visual observations from the environment; and the **VNM**, which estimates navigational affordances (distances between landmarks) and control actions. Each model is pre-trained on its own dataset, and we show that the complete system can execute a variety of user-specified instructions in real-world environments — choosing the correct sequence of landmarks by leveraging language and spatial context — and handle mistakes (such as missing landmarks). We also analyze the impact of each pre-trained model on the full system.

Limitations and future work. The most prominent limitation of LM-Nav is its reliance on landmarks: while the user can specify any instruction they want, LM-Nav only focuses on the landmarks and disregards any verbs, propositions, adverbs, etc. (e.g., "go straight for three blocks" or "drive past the dog very slowly"), which can be lossy. Grounding such nuances is an important direction for future work. Additionally, LM-Nav uses a **VNM** that is specific to outdoor navigation with the Jackal robot, which limits wider adoption for other robot embodiments and sensor suites. An exciting direction for future work would be to swap in a "general navigation model" [74] that can be utilized broadly across robots, analogous to how the **LLM** and **VLM** handle any text or image. In its current form, LM-Nav provides a simple and attractive prototype for how pre-trained models can be combined to solve complex robotic tasks, and illustrates that these models can serve as an "interface" to robotic controllers that are trained without any language annotations. One of the implications of this result is that further progress on self-supervised robotic policies (e.g., goal-conditioned policies) can directly benefit instruction following systems. More broadly, understanding how modern pre-trained models enable effective decomposition of robotic control may enable broadly generalizable systems in the future, and we hope that LM-Nav will serve as a step in this direction.

Acknowledgments

This research was supported by DARPA Assured Autonomy, DARPA RACER, Toyota Research Institute, ARL DCIST CRA W911NF-17-2-0181, and AFOSR. BO was supported by the Fulbright Junior Research Award granted by the Polish-U.S. Fulbright Commission. We would like to thank Alexander Toshev for pivotal discussions in early stages of the project. We would also like to thank Kuan Fang, Siddharth Karamcheti, Albertyna Osińska, Vijay Badrinarayanan, and Philip J. Ball for useful discussions and feedback.

References

[1] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sünderhauf, I. Reid, S. Gould, and A. van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018. 1, 2

[2] J. Gu, E. Stefani, Q. Wu, J. Thomason, and X. Wang. Vision-and-language navigation: A survey of tasks, methods, and future directions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022. 2

[3] A. Ku, P. Anderson, R. Patel, E. Ie, and J. Baldridge. Room-Across-Room: Multilingual vision-and-language navigation with dense spatiotemporal grounding. In *Conference on Empirical Methods for Natural Language Processing (EMNLP)*, 2020. 2

[4] V. Jain, G. Magalhaes, A. Ku, A. Vaswani, E. Ie, and J. Baldridge. Stay on the path: Instruction fidelity in vision-and-language navigation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[5] A. Yan, X. E. Wang, J. Feng, L. Li, and W. Y. Wang. Cross-lingual vision-language navigation, 2019. 1, 2

[6] T. Manderson, J. C. Gamboa, S. Wapnick, J. Tremblay, H. Zhao, F. Shkurti, D. Meger, and G. Dudek. Self-supervised, goal-conditioned policies for navigation in unstructured environments. 2010. 1, 3

[7] B. Sofman, E. L. Ratliff, J. A. D. Bagnell, J. Cole, N. Vandapel, and A. T. Stentz. Improving robot navigation through self-supervised online learning. *Journal of Field Robotics: Special Issue on Machine Learning Based Robotics in Unstructured Environments*, 2006.

[8] D. Gandhi, L. Pinto, and A. Gupta. Learning to fly by crashing. In *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017.

[9] A. Kouris and C.-S. Bouganis. Learning to fly by myself: A self-supervised cnn-based approach for autonomous navigation. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018.

[10] G. Kahn, A. Villaflor, B. Ding, P. Abbeel, and S. Levine. Self-Supervised Deep RL with Generalized Computation Graphs for Robot Navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2018.

[11] D. Shah, B. Eysenbach, G. Kahn, N. Rhinehart, and S. Levine. ViNG: Learning Open-World Navigation with Visual Goals. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2021. 1, 3

[12] T. Brown, B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, 2020. 1, 3, 4, 7

[13] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning*, 2021. 2, 3, 5, 7, 1

[14] P. Koehn. *Statistical Machine Translation*. Cambridge University Press, 2009. 2

[15] Y. W. Wong and R. Mooney. Learning for semantic parsing with statistical machine translation. In *Proceedings of the Human Language Technology Conference of the NAACL, Main Conference*, 2006. 2

[16] C. Matuszek, D. Fox, and K. Koscher. Following directions using statistical machine translation. In *2010 5th ACM/IEEE International Conference on Human-Robot Interaction (HRI)*, 2010.

[17] D. L. Chen and R. J. Mooney. Learning to interpret natural language navigation instructions from observations. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[18] S. Tellex, T. Kollar, S. Dickerson, M. R. Walter, A. G. Banerjee, S. Teller, and N. Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, 2011.

[19] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. *Learning to Parse Natural Language Commands to a Robot Control System*. 2013. 2

[20] N. Shimizu and A. Haas. Learning to follow navigational route instructions. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, IJCAI'09, 2009. 2

[21] H. Mei, M. Bansal, and M. R. Walter. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*, 2016. 2

[22] M. Shridhar, J. Thomason, D. Gordon, Y. Bisk, W. Han, R. Mottaghi, L. Zettlemoyer, and D. Fox. ALFRED: A Benchmark for Interpreting Grounded Instructions for Everyday Tasks. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[23] H. Chen, A. Suhr, D. Misra, N. Snavely, and Y. Artzi. Touchdown: Natural language navigation and spatial reasoning in visual street environments. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[24] K. M. Hermann, M. Malinowski, P. Mirowski, A. Banki-Horvath, K. Anderson, and R. Hadsell. Learning to follow directions in street view. *CoRR*, 2019.

[25] P. Mirowski, A. Banki-Horvath, K. Anderson, D. Teplyashin, K. M. Hermann, M. Malinowski, M. K. Grimes, K. Simonyan, K. Kavukcuoglu, A. Zisserman, and R. Hadsell. The streetlearn environment and dataset. *CoRR*, 2019.

[26] A. B. Vasudevan, D. Dai, and L. Van Gool. Talk2nav: Long-range vision-and-language navigation with dual attention and spatial memory. *Int. J. Comput. Vision*, 2021.

[27] D. K. Misra, A. Bennett, V. Blukis, E. Niklasson, M. Shatkhin, and Y. Artzi. Mapping instructions to actions in 3d environments with visual goal prediction. In E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[28] V. Blukis, N. Brukhim, A. Bennett, R. A. Knepper, and Y. Artzi. Following high-level navigation instructions on a simulated quadcopter with imitation learning. *CoRR*, 2018. 2

[29] J. Krantz, E. Wijmans, A. Majumdar, D. Batra, and S. Lee. Beyond the nav-graph: Vision-and-language navigation in continuous environments. In *Computer Vision – ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XXVIII*, 2020. 2

[30] P. Anderson, A. Shrivastava, J. Truong, A. Majumdar, D. Parikh, D. Batra, and S. Lee. Sim-to-real transfer for vision-and-language navigation. In *Proceedings of the 2020 Conference on Robot Learning*, 2021. 2

[31] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, and J. Brew. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, 2019. 2

[32] R. Thoppilan, D. D. Freitas, J. Hall, N. Shazeer, A. Kulshreshtha, H. Cheng, A. Jin, T. Bos, L. Baker, Y. Du, Y. Li, H. Lee, H. S. Zheng, A. Ghafouri, M. Menegali, Y. Huang, M. Krikun, D. Lepikhin, J. Qin, D. Chen, Y. Xu, Z. Chen, A. Roberts, M. Bosma, Y. Zhou, C. Chang, I. Krivokon, W. Rusch, M. Pickett, K. S. Meier-Hellstern, M. R. Morris, T. Doshi, R. D. Santos, T. Duke, J. Soraker, B. Zevenbergen, V. Prabhakaran, M. Diaz, B. Hutchinson, K. Olson, A. Molina, E. Hoffman-John, J. Lee, L. Aroyo, R. Rajakumar, A. Butryna, M. Lamm, V. Kuzmina, J. Fenton, A. Cohen, R. Bernstein, R. Kurzweil, B. Aguera-Arcas, C. Cui, M. Croak, E. H. Chi, and Q. Le. Lamda: Language models for dialog applications. *CoRR*, 2022.

[33] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code. *CoRR*, 2021. 2

[34] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents, 2022. 2

[35] C. Saharia, W. Chan, S. Saxena, L. Li, J. Whang, E. Denton, S. K. S. Ghasemipour, B. K. Ayan, S. S. Mahdavi, R. G. Lopes, T. Salimans, J. Ho, D. J. Fleet, and M. Norouzi. Photorealistic text-to-image diffusion models with deep language understanding, 2022.

[36] X. Gu, T.-Y. Lin, W. Kuo, and Y. Cui. Open-vocabulary object detection via vision and language knowledge distillation. In *International Conference on Learning Representations*, 2022. 7, 8

[37] C. Jia, Y. Yang, Y. Xia, Y.-T. Chen, Z. Parekh, H. Pham, Q. Le, Y.-H. Sung, Z. Li, and T. Duerig. Scaling up visual and vision-language representation learning with noisy text supervision. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[38] H. Song, L. Dong, W. Zhang, T. Liu, and F. Wei. CLIP models are few-shot learners: Empirical studies on VQA and visual entailment. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2022. 2

[39] M. Shridhar, L. Manuelli, and D. Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021. 2

[40] E. Jang, A. Irpan, M. Khansari, D. Kappler, F. Ebert, C. Lynch, S. Levine, and C. Finn. BC-z: Zero-shot task generalization with robotic imitation learning. In *5th Annual Conference on Robot Learning*, 2021. 2

[41] W. Huang, P. Abbeel, D. Pathak, and I. Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents. *arXiv preprint arXiv:2201.07207*, 2022. 2

[42] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. J. Ruano, K. Jeffrey, S. Jesmonth, N. Joshi, R. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan. Do as i can, not as i say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022. 2

[43] A. Zeng, M. Attarian, B. Ichter, K. Choromanski, A. Wong, S. Welker, F. Tombari, A. Purohit, M. Ryoo, V. Sindhwani, J. Lee, V. Vanhoucke, and P. Florence. Socratic models: Composing zero-shot multimodal reasoning with language. *arXiv*, 2022. 2

[44] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi. Simple but effective: Clip embeddings for embodied ai. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. 2

[45] D. Shah, B. Eysenbach, N. Rhinehart, and S. Levine. Rapid exploration for open-world navigation with latent goal models. In *5th Annual Conference on Robot Learning*, 2021. 3

[46] Manolis Savva*, Abhishek Kadian*, Oleksandr Maksymets*, Y. Zhao, E. Wijmans, B. Jain, J. Straub, J. Liu, V. Koltun, J. Malik, D. Parikh, and D. Batra. Habitat: A Platform for Embodied AI Research. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019. 3

[47] F. Xia, A. R. Zamir, Z.-Y. He, A. Sax, J. Malik, and S. Savarese. Gibson env: real-world perception for embodied agents. In *Computer Vision and Pattern Recognition (CVPR), 2018 IEEE Conference on*, 2018.

[48] M. Savva, A. X. Chang, A. Dosovitskiy, T. Funkhouser, and V. Koltun. MINOS: Multimodal indoor simulator for navigation in complex environments. *arXiv:1712.03931*, 2017.

[49] E. Kolve, R. Mottaghi, D. Gordon, Y. Zhu, A. Gupta, and A. Farhadi. AI2-THOR: an interactive 3d environment for visual AI. *CoRR*, 2017. 3

[50] A. Francis, A. Faust, H. T. L. Chiang, J. Hsu, J. C. Kew, M. Fiser, and T. W. E. Lee. Long-Range Indoor Navigation With PRM-RL. *IEEE Transactions on Robotics*, 2020. 3

[51] N. Hirose, F. Xia, R. Martín-Martín, A. Sadeghian, and S. Savarese. Deep visual MPC-policy learning for navigation. *IEEE Robotics and Automation Letters*, 2019. 3

[52] X. Meng, N. Ratliff, Y. Xiang, and D. Fox. Scaling Local Control to Large-Scale Topological Navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020. 3

[53] D. Shah and S. Levine. Viking: Vision-based kilometer-scale navigation with geographic hints. In *Robotics: Science and Systems (RSS)*, 2022. 3

[54] J.-B. Alayrac, J. Donahue, P. Luc, A. Miech, I. Barr, Y. Hasson, K. Lenc, A. Mensch, K. Millican, M. Reynolds, R. Ring, E. Rutherford, S. Cabi, T. Han, Z. Gong, S. Samangooei, M. Monteiro, J. Menick, S. Borgeaud, A. Brock, A. Nematzadeh, S. Sharifzadeh, M. Binkowski, R. Barreira, O. Vinyals, A. Zisserman, and K. Simonyan. Flamingo: a visual language model for few-shot learning, 2022. 3

[55] L. H. Li, M. Yatskar, D. Yin, C.-J. Hsieh, and K.-W. Chang. Visualbert: A simple and performant baseline for vision and language. In *Arxiv*, 2019.

[56] Y.-C. Chen, L. Li, L. Yu, A. E. Kholy, F. Ahmed, Z. Gan, Y. Cheng, and J. Liu. Uniter: Universal image-text representation learning. In *ECCV*, 2020. 3

[57] N. Savinov, A. Dosovitskiy, and V. Koltun. Semi-Parametric Topological Memory for Navigation. In *International Conference on Learning Representations*, 2018. 3

[58] D. S. Chaplot, D. Gandhi, S. Gupta, A. Gupta, and R. Salakhutdinov. Learning to Explore using Active Neural SLAM. In *International Conference on Learning Representations (ICLR)*, 2020.

[59] E. Wijmans, A. Kadian, A. Morcos, S. Lee, I. Essa, D. Parikh, M. Savva, and D. Batra. DD-PPO: Learning Near-Perfect PointGoal Navigators from 2.5 Billion Frames. In *International Conference on Learning Representations (ICLR)*, 2020. 3

[60] J. Bruce, N. Sunderhauf, P. Mirowski, R. Hadsell, and M. Milford. Learning deployable navigation policies at kilometer scale from a single traversal. In A. Billard, A. Dragan, J. Peters, and J. Morimoto, editors, *Proceedings of The 2nd Conference on Robot Learning*, 2018. 3

[61] L. P. Kaelbling. Learning to achieve goals. In *IJCAI*, pages 1094–1099, 1993. 4

[62] K. Hartikainen, X. Geng, T. Haarnoja, and S. Levine. Dynamical Distance Learning for Semi-Supervised and Unsupervised Skill Discovery. In *International Conference on Learning Representations*, 2020. 4

[63] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1959. 5

[64] M. Artetxe, S. Bhosale, N. Goyal, T. Mihaylov, M. Ott, S. Shleifer, X. V. Lin, J. Du, S. Iyer, R. Pasunuru, et al. Efficient large scale language modeling with mixtures of experts. *arXiv preprint arXiv:2112.10684*, 2021. 7

[65] B. Wang and A. Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. https://github.com/kingoflolz/mesh-transformer-jax, 2021. 7

[66] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang, M. Pieler, U. S. Prashanth, S. Purohit, L. Reynolds, J. Tow, B. Wang, and S. Weinbach. Gpt-neox-20b: An open-source autoregressive language model, 2022. 7

[67] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in Neural Information Processing Systems*, 2015. 7, 8

[68] M. Honnibal, I. Montani, S. Van Landeghem, and A. Boyd. spacy: Industrial-strength natural language processing in python. 2020. 7

[69] F. Rong. Extrapolating to unnatural language processing with gpt-3's in-context learning: The good, the bad, and the mysterious. http://ai.stanford.edu/blog/in-context-learning/, 2021. Accessed: 2022-06-04. 7, 1

[70] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 8

[71] T.-Y. Lin, M. Maire, S. J. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, 2014. 8

[72] Y. Wu, A. Kirillov, F. Massa, W.-Y. Lo, and R. Girshick. Detectron2. https://github.com/facebookresearch/detectron2, 2019. 8

[73] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, 2017. 8

[74] D. Shah, A. Sridhar, A. Bhorkar, N. Hirose, and S. Levine. GNM: A General Navigation Model to Drive Any Robot. In *arXiV*, 2022. URL https://arxiv.org/abs/2210.03370. 8