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

Anonymous Author(s)
Affiliation
Address
email

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 develop 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. We instantiate LM-Nav on a real-world mobile robot and demonstrate long-horizon navigation through complex, outdoor environments from natural language instructions.[1]

Keywords: instruction following, language models, vision-based navigation

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, generalizability, 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 self-supervised system for robotic navigation 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

[1]Please see the supplemental material for experiment videos and a Colab with the implementation.

Submitted to the 6th Conference on Robot Learning (CoRL 2022). Do not distribute.

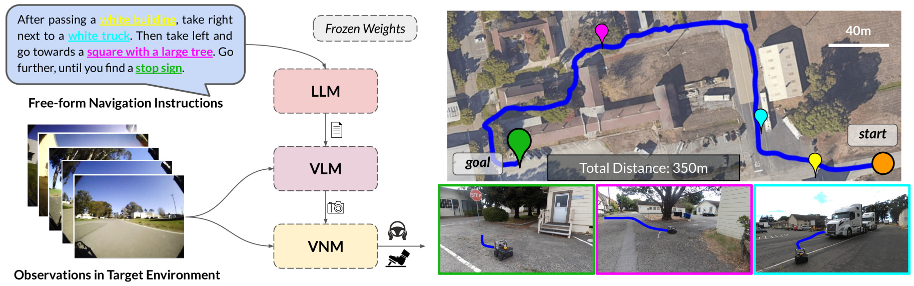

Figure 1: Embodied instruction following with LM-Nav: Our system takes as input a set of raw observations from the target environment and free-form textual instructions (left), deriving an actionable plan using three *pre-trained* models: a large language model (**LLM**) for extracting landmarks, a vision-and-language 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*.

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. 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 (**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 maximize a probabilistic objective, and find a plan for the robot, which is then executed by **VNM**.

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 self-supervised 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 new 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, Shridhar et al. [39], Khandelwal et al. [44] and Jang et al. [40] fine-tune embeddings from pre-trained models on robot data with language labels, Huang et al. [41] assume that the low-level agent can execute textual instructions (without addressing control), and Ahn et al. [42] assumes that the robot has a set of text-conditioned skills that can follow atomic textual commands. All of these approaches require access to low-level skills that can follow rudimentary textual commands, which in turn requires 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 in the target environment or 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. Notably, such a policy can be trained on large, diverse datasets and generalize to previously unseen environments [45, 52]. Being self-supervised, such policies are adept at navigating to desired goals specified by GPS locations or images, but are 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.

3 Preliminaries

LM-Nav consists of three large, independently pre-trained models for processing language, associating images with language, and associating images with robotic control and navigational affordances.

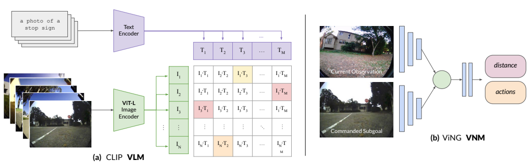

Figure 2: LM-Nav uses **VLM** to infer a joint probability distribution over textual landmarks and image observations. **VNM** constitutes an image-conditioned distance function and policy that can control the robot.

Large language models are generative models based on the Transformer architecture [53], trained on large corpora of internet text. LM-Nav uses the GPT-3 **LLM** [12], to parse textual instructions into a sequence of landmarks.

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 an embedding space that allows it to determine how likely some string is to be associated with a given image. We can 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 navigation behavior and navigational affordances directly from visual observations [11, 51, 57–60], 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). This provides an interface between images and embodiment. 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", comprising of a sequence of connected subgoals to a goal node, the **VNM** can navigate 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, 61, 62]. 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

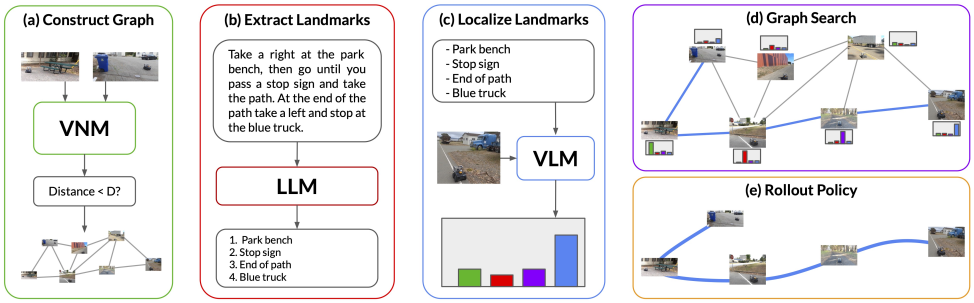

Figure 3: System overview: (a) **VNM** uses a goal-conditioned distance function to infer connectivity between the set of raw observations and constructs a topological graph. (b) **LLM** translates natural language instructions into a sequence of textual landmarks. (c) **VLM** infers a joint probability 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.

every connected pair of vertices $\{v_i, v_j\}$, we assign this distance estimate to the corresponding edge weight $D(v_i, v_j)$. For more details on the construction of this graph, see Appendix C.

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

LM-Nav combines the components discussed earlier to follow textual instructions in the real world. The **LLM** parses free-form instructions into a list of landmarks $\bar{l}$ (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 $\bar{l}$, $P_l(\bar{v}|\bar{l})$ (Sec. 4.3), and the **VNM** is then used to infer how effectively the robot can navigate between each pair of nodes in the graph, which we convert into a probability $P(\overline{v_i, v_j})$ derived from the estimated temporal distances. 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 optimal walk is then executed in the real world by using the actions produced by the **VNM** model.

4.1 Problem Formulation

We formulate the task of instruction following on the graph as that of maximizing the probability of successfully executing a walk that matches the instruction. As we will discuss in Section 4.2, we first parse the instruction into a list of landmarks $\bar{l} = l_1, l_2, \ldots, l_n$ that should be visited in order.[2] Recall that the **VNM** is used to build a topological graph that represents the connectivity of the environment from previously seen observations, with nodes $\{v_i\}$ corresponding to previously seen images. For a walk $\bar{v} = v_1, v_2, \ldots, v_T$, we factorize the probability that it corresponds to the given instruction into: (i) P_l, the probability that the walk visits all landmarks from the description; (ii) P_t, the probability that the walk $\bar{v}$ can be executed successfully. Let $\bar{l} = l_1, l_2, \ldots, l_n$ be the list of landmarks described in the natural language instructions, and let $P(l_i|v_j)$ denote the probability that node v_j corresponds to the landmark description l_i. Then we have:

$$P_l(\bar{v}|\bar{l}) = \max_{1 \le t_1 \le t_2 \le \ldots \le t_n \le T} \prod_{1 \le k \le n} P(l_k|v_{t_k}), \tag{1}$$

where $t_1, t_2, \ldots, t_n$ is assignment of a subsequence of walk's node to landmark descriptions.

To obtain the probability $P_t(\bar{v})$, we must convert the distance estimates provided by the **VNM** model into probabilities. This has been studied in the literature on goal-conditioned policies [63, 64]. A simple model based on a discounted MDP formulation is to model the probability of successfully reaching the goal as γ to the power number of time steps, which corresponds to a probability of termination of $1 - \gamma$ at each time step. We then have

$$P_t(\bar{v}) = \prod_{1 \le j < n} P(\overline{v_j, v_{j+1}}) = \prod_{1 \le j < n} \gamma^{D(v_j, v_{j+1})}, \tag{2}$$

[2] LM-Nav discards any such information beyond landmarks (e.g. verbs), and this represents a limitation of our approach. Incorporating more nuanced commands is an important direction for future work.

where $D(v_j, v_{j+1})$ refers to the length (in the number of time steps) of the edge between nodes v_j and v_{j+1}, which is provided by the **VNM** model. The final probabilistic objective that our system needs to maximize becomes:

$$P_M(\bar{v}) = P_t(\bar{v})P_l(\bar{v}|\bar{l}) = \prod_{1\leq j<n} \gamma^{D(v_j,v_{j+1})} \max_{1\leq t_1\leq t_2\leq\ldots\leq t_n\leq t} \prod_{1\leq k\leq n} P(l_k|v_{t_k}). \tag{3}$$

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 standard large language model, which in our prototype is GPT-3 [12]. We used a prompt with 3 examples of correct landmarks' extractions, followed up by the description to be translated by the **LLM**. Such an approach worked for the instructions that we tested it on. Examples of instructions together with landmarks extracted by the model can be found in Fig. 4. The appropriate selection of the prompt, including those 3 examples, was required for more nuanced cases. 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 $P(l_i|v_j)$, the probability that landmark description l_i refers to node v_j (see Equation 1). 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, to employ CLIP to compute $P(l_i|v_j)$, we use the image at node v_j and caption prompts in the form of *"This is a photo of a [l_i]"*. The resulting probability $P(l_i|v_j)$, together with the inferred edges' distances will be used to select the optimal walk in the graph.

4.4 Graph Search for the Optimal Walk

As described in Sec. 4.1, LM-Nav aims at finding a walk $\bar{v} = (v_1, v_2, \ldots, v_T)$ that maximizes the probability of successful execution that adheres to the given instructions. We formalized this probability P_M defined by Eqn. 3. We can define a function $R(\bar{v}, \bar{t})$ for a monotonically increasing sequence of indices $\bar{t} = (t_1, t_2, \ldots, t_n)$:

$$R(\bar{v}, \bar{t}) := \sum_{i=1}^{n} \log P(l_i|v_{t_i}) - \alpha \sum_{j=1}^{T-1} D(v_j, v_{j+1}), \text{where } \alpha = -\log\gamma. \tag{4}$$

which has the property that $(\bar{v})$ maximizes P_M 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 $i \in \{0, 1, \ldots, n\}$, $v \in V$:

$$Q(i, v) = \max_{\substack{\bar{v}=(v_1,v_2,\ldots,v_j),v_j=v \\ \bar{t}=(t_1,t_2,\ldots,t_i)}} R(\bar{v}, \bar{t}). \tag{5}$$

$Q(i, v)$ 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 node S. For $i > 0$ we have:

$$Q(i, v) = \max\left(Q(i-1, v) + \log P(l_i|v), \max_{w\in\text{neighbors}(v)} Q(i, w) - \alpha \cdot D(v, w)\right). \tag{6}$$

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 ([65]). In each iteration, we pick the node v with the largest value of $Q(i, v)$ and update its neighbors based on the Eqn. 6. Algorithm 1 summarizes this search process. The result of this algorithm is a walk $\bar{v} = (v_1, v_2, \ldots, v_T)$ that maximizes the probability of successfully carrying out the instruction. Given such a walk, **VNM** can execute the path by using its action estimates to sequentially navigate to these nodes.

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). LM-Nav can follow instructions *over 100s of meters* and visits all specified landmarks except a fire hydrant (c).

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. 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.

Algorithm 1: Graph Search

Input: Landmarks $(l_1, l_2, \ldots, l_n)$.
Input: Graph $\mathcal{G}(V, E)$.
Input: Starting node S.
$\forall_{\substack{i=0,\ldots,n \\ v \in V}} \; Q[l_i, v] = -\infty$
$Q[0, S] = 0$
Dijkstra_algorithm$(\mathcal{G}, Q[0, *])$
for i in $1, 2, \ldots, n$ **do**
  $\forall_{v \in V} Q[i, v] = Q[i-1, v] + \text{CLIP}(l_i, v)$
  Dijkstra_algorithm$(\mathcal{G}, Q[i, *])$
end for
destination $= \arg\max(Q[n, *])$
return backtrack(destination, $Q[n, *]$)

5.1 Mobile Robot Platform

We implement LM-Nav on a Clearpath Jackal UGV platform (see Fig. 1(right)). The sensor suite consists of a 6-DoF IMU, a GPS unit for approximate localization, wheel encoders for local odometry, and front- and rear-facing RGB cameras with a 170° field-of-view for capturing visual observations and localization in the topological graph. The **LLM** and **VLM** queries are pre-computed on a remote workstation and the computed path is commanded to the robot wirelessly. The **VNM** runs on-board and only uses forward RGB images and unfiltered GPS measurements.

5.2 Following Instructions with LM-Nav

In each evaluation scene, we first construct the graph by manually driving the robot and collecting image and GPS observations. The graph is constructed automatically using the **VNM** from this data, and in principle such data could also be obtained from past traversals, or even with autonomous exploration methods [45]. Once the graph is constructed, the robot can carry out instructions in that environment. We tested our system on a total of 5 queries (presented in Fig. 4,5), corresponding to a total combined length of about 2 km. Out of the 19 landmarks, LM-Nav correctly visited all but one. This mistake is attributed to the failure of detecting the landmark by the VLM (See *Missing landmarks* below). We did not observe any issues with **LLM**, **VNM**, or the graph search algorithm.

Fig. 4 shows qualitative examples of the path taken by the robot, along with the number of landmarks that are visited successfully in the right order; note that the overhead image and spatial localization

LLM Candidate	Parsing Success
Noun Chunks	0.79
GPT-2 [66]	0.48
GPT-J-6B [67]	0.70
GPT-3 [12] (Ours)	**1.0**

Table 1: GPT-3 consistently outperforms alternatives in parsing free-form instructions into landmarks.

VLM Candidate	Detection Rate
Faster-RCNN [68]	0.07
ViLD [36]	0.38
CLIP-ViT [13] (Ours)	**0.87**

Table 2: CLIP-ViT produces the most reliable landmark detections from visual observations.

of the landmarks is *not* available to the robot and is shown for visualization only. 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. 3 causes the robot to pick the correct stop sign in context, so as to minimize overall travel distance. Fig. 4(b) highlights LM-Nav's ability to parse complex instructions with multiple landmarks specifying the route — despite the possibility of a shorter route directly to the final landmark that ignores instructions, the robot finds a path that visits all of the landmarks in the correct order.

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.

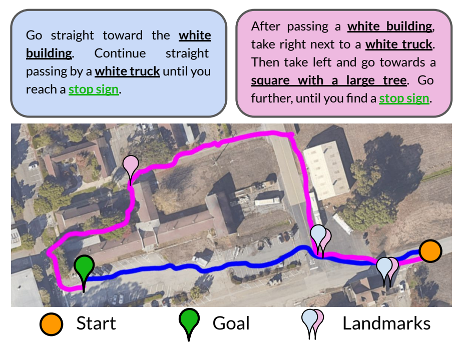

Figure 5: LM-Nav can successfully disambiguate instructions with same start-goal locations that differ slightly, and execute them. Extracted landmarks and their corresponding locations are highlighted and marked with a pin, respectively.

Missing landmarks. While LM-Nav is effective at parsing landmarks from instructions, localizing them on the graph, and finding a path to the goal, 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 our 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.

5.3 Dissecting LM-Nav

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

We evaluated the performance of different methods at extracting *ordered* list of landmarks given a free-form instruction. We compare GPT-3 used by LM-Nav to alternative pre-trained transformer models — GPT-2 [66] and GPT-J-6B [67] — and a simple baseline using spaCy NLP library [69] that extracts base noun phrases and filters out certain words (e.g.: *you*, *right*). We report the average number of correctly extracted landmarks in Table 1. GPT-3 significantly outperforms other models, owing to its superior capacity and in-context learning [70]. Surprisingly, noun chunking performs reliably in small, direct prompts (e.g. Fig. 4(a)). For further details on these experiments and prompt engineering for the models, see Appendix A.

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 experiments from Sec. 5.2 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 credible object detection alternatives — Faster-RCNN-FPN [68, 71], a state-of-the-art object detection model pre-trained on MS-COCO [72, 73], and ViLD [36], an open-vocabulary object detector based on CLIP and Mask-RCNN [74]. 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 2). 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. Fig. 6 shows that, while such a controller works well on open roads, it cannot 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 enabling LM-Nav to navigate in complex environments without collisions.

Figure 6: LM-Nav with a GPS-only controller fails to execute a plan due to its inability to reason about traversability through obstacles.

6 Discussion

We presented **L**arge **M**odel **Nav**igation, or LM-Nav, a system for robotic navigation from textual instructions that can control a mobile robot without requiring any user annotations for navigational data. LM-Nav combines three pre-trained models: the **LLM**, which parses user instructions into a list of landmarks, the **VLM**, which estimates the probability that each observation in a "mental map" constructed from prior exploration of the environment matches these landmarks, and the **VNM**, which estimates navigational affordances (distances between landmarks) and robot 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 outdoor environments — choosing the correct sequence of landmarks through a combination of 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 or other commands (e.g., "go straight for three blocks" or "drive past the dog very slowly"). Grounding verbs and other nuanced commands is an important direction for future work. Additionally, LM-Nav uses a **VNM** that is specific to outdoor navigation with the Clearpath Jackal robot. An exciting direction for future work would be to design a more general "large navigation model" that can be utilized broadly on any robot, analogous to how the **LLM** and **VLM** handle any text or image. However, we believe that 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 our work will serve as a step in this direction.

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)*, pages 7606–7623, 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*, pages 1862–1872, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1181. URL `https://aclanthology.org/P19-1181`.

[5] A. Yan, X. E. Wang, J. Feng, L. Li, and W. Y. Wang. Cross-lingual vision-language navigation, 2019. URL `https://arxiv.org/abs/1910.11301`. 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*, 23(12):1059 – 1075, December 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)*, pages 3948–3955, 2017. doi:10.1109/IROS.2017.8206247.

[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)*, pages 1–9, 2018. doi:10.1109/IROS.2018.8594204.

[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, 2, 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 H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL `https://proceedings.neurips.cc/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf`. 2, 3, 5, 7, 1

[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*, pages 8748–8763. PMLR, 2021. 2, 3, 5, 7, 1

[14] P. Koehn. *Statistical Machine Translation*. Cambridge University Press, 2009. doi:10.1017/CBO9780511815829. 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*, pages 439–446, New York City, USA, June 2006. Association for Computational Linguistics. URL `https://aclanthology.org/N06-1056`. 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)*, pages 251–258, 2010. doi:10.1109/HRI.2010.5453189.

[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*, AAAI'11, page 859–865. AAAI Press, 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*, AAAI'11, page 1507–1514. AAAI Press, 2011.

[19] C. Matuszek, E. Herbst, L. Zettlemoyer, and D. Fox. *Learning to Parse Natural Language Commands to a Robot Control System*, pages 403–415. Springer International Publishing, Heidelberg, 2013. ISBN 978-3-319-00065-7. doi:10.1007/978-3-319-00065-7_28. URL `https://doi.org/10.1007/978-3-319-00065-7_28`. 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, page 1488–1493, San Francisco, CA, USA, 2009. Morgan Kaufmann Publishers Inc. 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. URL `https://arxiv.org/abs/1912.01734`. 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)*, pages 12530–12539, 2019. doi:10.1109/CVPR.2019.01282. 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. doi:10.48550/ARXIV.1903.00401. URL `https://arxiv.org/abs/1903.00401`.

[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*, abs/1903.01292, 2019. URL `http://arxiv.org/abs/1903.01292`.

[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*, 129(1):246–266, jan 2021. ISSN 0920-5691. doi:10.1007/s11263-020-01374-3. URL https://doi.org/10.1007/s11263-020-01374-3.

[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, Brussels, Belgium, October 31 - November 4, 2018*, pages 2667–2678. Association for Computational Linguistics, 2018. doi:10.18653/v1/d18-1287. URL https://doi.org/10.18653/v1/d18-1287.

[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*, abs/1806.00047, 2018. URL http://arxiv.org/abs/1806.00047. 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*, page 104–120, Berlin, Heidelberg, 2020. Springer-Verlag. ISBN 978-3-030-58603-4. URL https://doi.org/10.1007/978-3-030-58604-1_7. 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 J. Kober, F. Ramos, and C. Tomlin, editors, *Proceedings of the 2020 Conference on Robot Learning*, volume 155 of *Proceedings of Machine Learning Research*, pages 671–681. PMLR, 16–18 Nov 2021. URL https://proceedings.mlr.press/v155/anderson21a.html. 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*, abs/1910.03771, 2019. URL http://arxiv.org/abs/1910.03771. 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*, abs/2201.08239, 2022. URL https://arxiv.org/abs/2201.08239.

[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*, abs/2107.03374, 2021. URL https://arxiv.org/abs/2107.03374. 2

[34] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with clip latents, 2022. URL https://arxiv.org/abs/2204.06125. 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. URL `https://arxiv.org/abs/2205.11487`.

[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. URL `https://openreview.net/forum?id=lL3lnMbR4WU`. 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 M. Meila and T. Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 4904–4916. PMLR, 18–24 Jul 2021. URL `https://proceedings.mlr.press/v139/jia21b.html`.

[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)*, pages 6088–6100, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi:10.18653/v1/2022.acl-long.421. URL `https://aclanthology.org/2022.acl-long.421`. 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. 3

[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. URL `https://openreview.net/forum?id=8kbp23tSGYv`. 3

[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. 3

[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. 3

[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)*. IEEE, 2022. URL `https://arxiv.org/abs/2111.09888`. 3

[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, 6

[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*. IEEE, 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*, abs/1712.05474, 2017. URL http://arxiv.org/abs/1712.05474. 3

[50] A. Faust, K. Oslund, O. Ramirez, A. Francis, L. Tapia, M. Fiser, and J. Davidson. PRM-RL: Long-range Robotic Navigation Tasks by Combining Reinforcement Learning and Sampling-Based Planning. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5113–5120, 2018. doi:10.1109/ICRA.2018.8461096. 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] D. Shah and S. Levine. Viking: Vision-based kilometer-scale navigation with geographic hints. In *Robotics: Science and Systems (RSS)*, 2022. URL https://arxiv.org/abs/2202.11271. 3

[53] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf. 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. URL https://arxiv.org/abs/2204.14198. 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] C. Chen, A. Seff, A. Kornhauser, and J. Xiao. Deepdriving: Learning affordance for direct perception in autonomous driving. In *IEEE International Conference on Computer Vision*, 2015. 3

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

[59] 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.

[60] 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

[61] 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

[62] 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

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

[64] 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

[65] E. W. Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1 (1):269–271, 1959. 5

[66] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 7, 2

[67] B. Wang and A. Komatsuzaki. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`, May 2021. 7, 2

[68] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In C. Cortes, N. Lawrence, D. Lee, M. Sugiyama, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 28. Curran Associates, Inc., 2015. URL `https://proceedings.neurips.cc/paper/2015/file/14bfa6bb14875e45bba028a21ed38046-Paper.pdf`. 7, 8

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

[70] 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

[71] 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)*, pages 936–944, 2017. doi:10.1109/CVPR.2017.106. 8

[72] 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

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

[74] K. He, G. Gkioxari, P. Dollár, and R. Girshick. Mask r-cnn. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017. doi:10.1109/ICCV.2017.322. 8

Part I

Appendix

Table of Contents

A Prompt Engineering

To use large language models for a particular task, as opposed to a general text completion, one needs to encode the task as a part of the text input to the model. There exist many ways to create such encoding and the process of the representation optimization is sometimes referred to as *prompt engineering* [13]. In this section, we discuss the prompts we used for **LLM** and **VLM**.

A.1 LLM Prompt Engineering

All our experiments use GPT-3 [12] as the **LLM**, accessible via OpenAI's API: `https://openai.com/api/`. We used this model to extract a list of landmarks from free-form instructions. The model outputs were very reliable and robust to small changes in the input prompts. For parsing simple queries, GPT-3 was surprisingly effective with a single, zero-shot prompt. See the example below, where the model output is highlighted:

```
First, you need to find a stop sign. Then take left and
right and continue until you reach a square with a tree.
Continue first straight, then right, until you find a white
truck. The final destination is a white building.
Landmarks:
1. Stop sign
2. Square with a tree
3. White truck
4. White building
```

While this prompt is sufficient for simple instructions, more complex instructions require the model to reason about occurrences such as re-orderings, e.g. *Look for a glass building after after you pass by a white car*. We leverage GPT-3 ability to perform *in-context learning* [70] by adding three examples in the prompt, along with the word `Ordered`:

```
Take right next to an old white building. Look for a fire
station, which you will see after passing by a school.
```

```
Ordered landmarks:
1.  an old white building
2.  a school
3.  a fire station

Go straight for two blocks.  Take right at a roundabout,
before it you will pass a big, blue tree.
Ordered landmarks:
1.  a big, blue tree
2.  a roundabout

Look for a library, after taking a right turn next to a
statue.
Ordered landmarks:
1.  a statue
2.  a library

[Instructions]
Ordered landmarks:
1.  ...
```

We use the above prompt in all our experiments (Section 5.2 and Appendix B), and GPT-3 was successfully able to extract landmarks with a parsing success of 98%. For the ablation experiments described in Section 5.3 we have discovered that GPT-2 [66] and GPT-J-6B [67] work better with the first, zero-shot prompt.

A.2 VLM Prompt Engineering

In the case of our **VLM**— CLIP [13] — we use a simple family of prompts: *This is a photo of* ___, appended with the landmark description. This simple prompt was sufficient to detect over 95% of the landmarks encountered in our experiments. While our experiments did not require more careful prompt engineering, Radford et al. [13] and Zeng et al. [43] report improved robustness by using an ensemble of slightly varying prompts.

B Quantitative Analysis of LM-Nav's Performance

This section presents a quantitative analysis of LM-Nav's performance in complex, real-world environments. Following the recipe outlined in Section 5.2, we evaluate our system in two environments of varying scale and complexity by providing 10 instructions in each of them. For instructions, we chose a set of prominent landmarks in the environment that can be identified from the robot's low-resolution camera observations, e.g. traffic cones, cars, stop signs, etc.

To better quantify the performance of LM-Nav, we introduce some performance metrics. A walk produced by the graph search is considered *successful*, if (1) it matches the path intended by the user or (2) if the landmark images extracted by the search algorithm indeed contain said landmarks (i.e. if the produced path is *valid*, if not identical). The fraction of successful walks produced by the search algorithm is defined as *planning success*. For a successfully executed plan in the real world, we define *efficiency* as:

$$\min(1, \frac{\text{length of described route}}{\text{length of executed route}}).$$

The second term — corresponding to the optimality of the executed route — is clipped at a maximum of 1 to account for occasional cases when the **VNM** executes a shorter, more direct path than the user intended. For a set of queries, we report the average efficiency over successful experiments.

Environment	Expt. Length (m)	Efficiency ↑	# Diseng. ↓	Planning Success ↑
`EnvSmall-10`	168.2	0.96	0.1	0.9
`EnvLarge-10`	470.4	0.89	0	0.8

Table 3: 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.

System	Net Success ↑	Efficiency ↑	# Diseng. ↓	Planning Success ↑
GPS-Nav (No **VNM**)	23%	**0.93**	0.75	**90%**
LM-Nav (Ours)	**88%**	**0.91**	**0.1**	**90%**

Table 4: Ablating the navigation model **VNM**, we see that a naïve GPS low-level controller is unable to reason about obstacles and traversability, frequently resulting in collisions or disengagements.

The *planning efficiency* is analogously defined as:

$$\min(1, \frac{\text{length of described route}}{\text{length of planned walk}}).$$

Yet another metric — *the number of disengagements* — counts the average number of human interventions required per experiment, due to unsafe maneuvers like collisions or falling off a curb, etc.

Table 3 summarizes the quantitative performance of the system over 20 instructions. LM-Nav can consistently follow the instructions in 85% of the experiments, without collisions or disengagements (an average of 1 intervention per 6.4km of traversals). In all the unsuccessful experiments, the failure can be attributed to the inability of the planning stage — the search algorithm is unable to visually localize certain "hard" landmarks in the graph — leading to incomplete execution of the instructions. Investigating these failure modes suggests that the performance of our system is bottlenecked by the ability of **VLM** to detect unfamiliar landmarks, e.g. a fire hydrant, and in challenging lighting conditions, e.g. underexposed images.

As a baseline, we also report these performance metrics with an ablation of our system that replaces the **VNM** with GPS-based distance estimates and a naïve bee-lining controller (see Section 5.3 for further discussion on this ablation). Table 4 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. LM-Nav can leverage the strengths of all three pre-trained models to successfully follow instructions over large distances without disengagements.

C Building the Topological Graph with VNM

This section outlines finer details regarding how the topological graph is constructed using **VNM**. We use a combination of learned distance estimates (from **VNM**), spatial proximity (from GPS), and temporal proximity (during data collection), to deduce edge connectivity. If the corresponding timestamps of two nodes are close ($< 2s$), suggesting that they were captured in quick succession, then the corresponding nodes are connected — adding edges that were physically traversed. If the **VNM** estimates of the images at two nodes are close, suggesting that they are *reachable*, then the corresponding nodes are also connected — adding edges between distant nodes along the same route and giving us a mechanism to connect nodes that were collected in different trajectories or at different times of day but correspond to the nearby locations. To avoid cases of underestimated distances by the model due to aliased observations, e.g. green open fields or a white wall, we filter out prospective edges that are significantly further away as per their GPS estimates — thus, if two nodes are nearby as per their GPS, e.g. nodes on different sides of a wall, they may not be disconnected if the **VNM** does not estimate a small distance; but two similar-looking nodes 100s of meters away, that may be facing a white wall, may have a small **VNM** estimate but are not added to the graph to avoid *wormholes*. Algorithm 2 summarizes this process — the timestamp threshold ϵ is 1 second, the learned distance threshold τ is 80 time steps (corresponding to $\sim$ 20 meters), and the spatial threshold η is 100 meters.

Algorithm 2: Graph Building

1: **Input**: Nodes $n_i, n_j \in \mathcal{G}$ containing robot observations; **VNM** distance function f_d; hyperparameters $\{\tau, \epsilon, \eta\}$
2: **Output**: Boolean e_{ij} corresponding to the existence of edge in $\mathcal{G}$, and its weight
3: learned distance $D_{ij} = f_d(n_i[\text{'image'}], n_j[\text{'image}'])$
4: timestamp distance $T_{ij} = |n_i[\text{'timestamp'}] - n_j[\text{'timestamp'}]|$
5: spatial distance $X_{ij} = \|n_i[\text{'GPS'}] - n_j[\text{'GPS'}])\|$
6: **if** ($T_{ij} < \epsilon$) **then** return $\{$*True,* $D_{ij}\}$
7: **else if** $(D_{ij} < \tau)$ AND $(X_{ij} < \eta)$ **then** return $\{$*True,* $D_{ij}\}$
8: **else** return *False*

Since a graph obtained by such an analysis may be quite dense, we perform a *transitive reduction* operation on the graph to remove redundant edges.

D Miscellaneous Ablation Experiments

D.1 Ablating the Search Objective

The graph search objective described in Section 4.4 can be factored into two components: visiting the required landmarks (denoted by $P_l(\bar{v}|\bar{l})$) and minimizing distance traveled (denoted by $P_t(\bar{v})$). To analyze the importance of these two components, we ran a set of experiments where the nodes to be visited are selected based only on P_l. This corresponds to a *Max Likelihood* planner, which only picks the most likely node for each landmark, without reasoning about their relative topological positions and traversability. This approach leads to a simpler algorithm: for each of the landmark descriptions, the algorithm selects the node with the highest CLIP score and connects it via the shortest path to the current node. The shortest path between each pair of nodes is computed using the Floyd–Warshall algorithm.

Table 5 summarizes the performance metrics for the two planners. Unsurprisingly, the max likelihood planner suffers greatly in the form of efficiency, because it does not incentivize shorter paths (see Figure 7 for an example). Interestingly, the planning success suffers as well, especially in complex environments. Further analysis of these failure modes reveals cases where **VLM** returns erroneous detections for some landmarks, likely due to the contrastive objective struggling with variable binding (see Figure 8 for an example). While LM-Nav suffers from these failures as well, the second factor in the search objective $P_t(\bar{v})$ imposes a *soft constraint* on the search space of the landmarks, eliminating most of these cases and resulting in a significantly higher planning success rate.

	`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	0.9	0.80	0.8	0.99

Table 5: Comparison of the planning success and planning efficiency of LM-Nav and its modification selecting nodes only based on the best CLIP score.

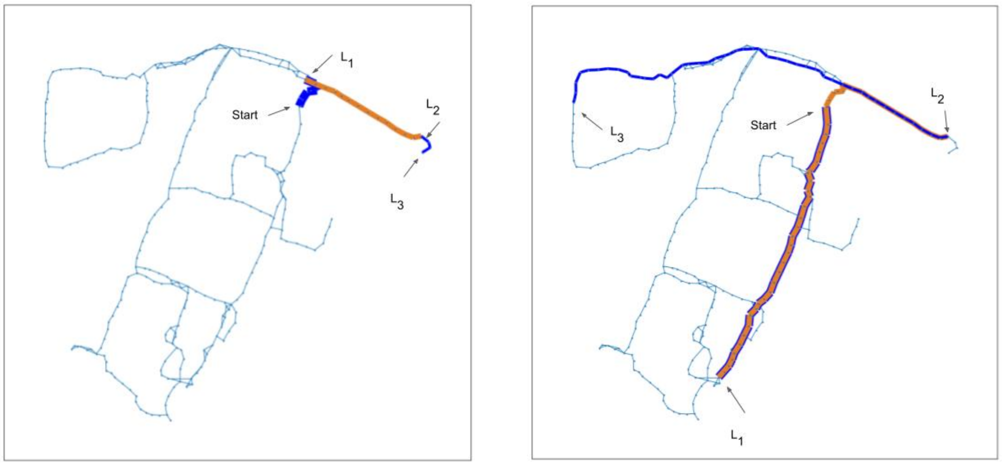

Figure 7: Examples of path planned by LM-Nav (left) and maximum likelihood planning (right). The start nodes and detected nodes are indicated with black arrows. In order to represent overlapping paths, we use colors interchangeably (start $\rightarrow L_1$: blue, $L_1 \rightarrow L_2$: orange, $L_2 \rightarrow L_3$: blue). The path taken by LM-Nav is significantly shorter, resulting in a $5\times$ more efficient plan.

Figure 8: An example of failure to pick the correct image by maximum likelihood planning. Both images were selected for a prompt *A photo of a blue dumpster*. The left one was selected as a part of the LM-Nav's graph search and the right was selected by maximum likelihood planning. In the latter case, the selected image contains a blue semi-truck and an orange trailer, but no blue dumpsters. This might be an example of an issue with the variable binding. The left image was edited to maintain anonymity.

E Interim Code Release

We are sharing the code corresponding to the **LLM** interface, **VLM** scoring, and graph search algorithm — along with a user-friendly Jupyter notebook capable of running quantitative experiments from Section B. The code is available in the supplemental material (please see folder `lmnav_code_release/`). Due to upload size constraints, the pickled graph objects can be found at our project page: https://sites.google.com/view/lmnav-anon.

F Experiment Videos

We are sharing experiment videos of LM-Nav deployed on a Clearpath Jackal mobile robotic platform — please see `lmnav_video.mp4` in the supplemental material. The videos highlight the behavior learned by LM-Nav for the task of following free-form textual instructions and its ability to navigate complex environments and disambiguate between fine-grained commands. A higher resolution video is also available at the project page: https://sites.google.com/view/lmnav-anon.