Designing LLM based agents to interact with the embodied world

Dylan Goetting

Electrical Engineering and Computer Sciences
University of California, Berkeley

Technical Report No. UCB/EECS-2025-59
http://www2.eecs.berkeley.edu/Pubs/TechRpts/2025/EECS-2025-59.html

May 14, 2025

Acknowledgement

I would like to thank my collaborators in the Malik group, including Antonio Loquercio, Andrea Bajcsy, Himanshu Gaurav Singh, Haoran Geng who have provided me with generous guidance throughout my academic career. I'm grateful for both Jitendra Malik and Dawn Song for their oversight. I also thank Daniel Flaherty, Ana Cismaru and Tarun Amarnath for helping guide my process through graduate school and my career.

Designing LLM based agents to interact with the embodied world

by

Dylan Goetting

A thesis submitted in partial satisfaction of the

requirements for the degree of

Masters of Science, Plan II

in

Electrical Engineering and Computer Science

in the

Graduate Division

of the

University of California, Berkeley

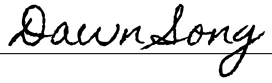
Committee in charge:

Professor Jitendra Malik, Chair
Professor Dawn Song, Co-chair

Spring 2025

The thesis of Dylan Goetting, titled Designing LLM based agents to interact with the embodied world, is approved:

Chair  Date 5/14/25

Co-chair  Date 5/14/25

University of California, Berkeley

Designing LLM based agents to interact with the embodied world

Abstract

Designing LLM based agents to interact with the embodied world

by

Dylan Goetting

Masters of Science, Plan II in Electrical Engineering and Computer Science

University of California, Berkeley

Professor Jitendra Malik, Chair

Professor Dawn Song, Co-chair

Large Language Models (LLMs) have seen rapid advancements across different modalities, yet they remain mostly isolated from physical environments. Meanwhile, robotics research continues to face challenges in generalization and scalability, limited by costly and narrow data collection processes. In this work, we study methods to bridge the gap between LLMs and physical robotic systems through structured observation and action interfaces. We first introduce *VLMnav*[1], a novel framework that transforms a Vision-Language Model (VLM) into an end-to-end navigation policy, allowing it to select low-level actions directly from visual input without fine-tuning. We evaluate its navigation capabilities on multiple benchmarks and perform a detailed design analysis. Building on this, we extend to a more complex manipulation setting, where the agent calls a Vision-Language-Action (VLA) model to handle fine-grained control. We analyze both task performance and design factors and show how the agent can most effectively utilize the capabilities of the VLA.

[1]This chapter is based on [14], in collaboration with Himanshu Gaurav Singh and Antonio Loquercio

To my parents

Contents

Contents ii

List of Figures iii

List of Tables v

1 Introduction 1
- 1.1 LLMs and Robotics . . . 1
- 1.2 Designing Embodied Agents . . . 3

2 Navigation 4
- 2.1 Introduction . . . 4
- 2.2 Related Work . . . 5
- 2.3 Overview . . . 7
- 2.4 Experiments . . . 10
- 2.5 Conclusion . . . 15

3 Manipulation 17
- 3.1 Introduction . . . 17
- 3.2 Related Work . . . 17
- 3.3 VLA as Tools . . . 19

Bibliography 24

List of Figures

1.1 **LLMs as embodied agents:** In our navigation framework, the LLM directly selects low-level actions based on visual input, whereas our manipulation framework delegates fine-grained actions to a specialized VLA tool, yielding a higher-level abstraction. 2

2.1 **Prompt:** The full action prompt for VLMnav consists of three parts: A system prompt to describe the embodiment, an action prompt to describe the task, the potential actions, and the output instruction, and an image prompt showing the current observation along with the annotated actions 5

2.2 **Approach:** Our method is made up of four key components: (i) *Navigability*, which determines locations the agent can actually move to, and updates the voxel map accordingly. An example update step to the map shows the marking of new area as explored (gray) or unexplored (green). (ii) *Action Proposer*, which refines a set of final actions according to spacing and exploration. (iii) *Projection*, which visually annotates the image with actions. (iv) *Prompting*, which constructs a detailed chain-of-thought prompt to select an action. 6

2.3 **Navigability:** An example step of the *Navigability* subroutine. The navigability mask is shown in blue and polar actions making up A_{initial} are in green 8

2.4 **Prompting:** The separate prompt for determining episode termination 10

2.5 **Baselines:** Comparing the four different methods on a sample image. *Ours* contains arrows that point to navigable locations, *PIVOT* has arrows sampled from a random 2-D Gaussian, *Ours w/o nav* sees uniformly spaced arrows (note arrows 3 and 5 point into a wall), and *Prompt Only* sees just the raw RGB image 11

2.6 **Impact of sensor FOV:** We evaluate the performance of four different sensor FOVs, and find that a wider FOV invariably leads to higher performance 14

3.1 **Robotic generalists require several layers of abstraction:**. VLAs can generalize across tasks, but they fall short of the high-level planning capabilities of LLMs. Works shown in blue bridge this gap by allowing LLMs to solve tasks by calling low-level robotic APIs. This work builds off of those ideas by connecting LLMs to VLAs, enabling a higher level of abstraction . 18

3.2 **Overview of the proposed framework:** In one process, the VLA continuously executes an instruction in its environment. Concurrently, a central planning agent chooses what instructions to send to the VLA, and a feedback module periodically sends updates to the planning agent. The trajectory results are stored offline for future prompting of the central agent 19

3.3 **Language sensitivity analysis:** OpenVLA is very sensitive to the specific phrasing used in the instruction 21

List of Tables

2.1 **ObjectNav Results:** We evaluate four different prompting strategies on the ObjectNav benchmark, and see our method achieves highest performance in both accuracy (SR) and efficiency (SPL). Ablating the *allow_slide* parameter shows our method is dependent on sliding past obstacles 12

2.2 **GOAT Results:** Comparison of prompting strategies on GOAT Bench, a more challenging navigation task. Across three different goal modalities, our method strongly outperforms baseline methods 12

2.3 **GOAT comparison:** Directly comparing to other works, we see that specialized systems still produce superior performance. We also note these other works use a narrower FOV, lower image resolution, and a different action space, which could explain some of the differences . . 13

2.4 **Impact of adding context history:** We compare our method to alternatives of keeping the past 0, 5, 10, and 15 observations and actions. We see that adding context history does not improve the performance of our method 14

2.5 **Depth Ablation:** We evaluate two alternate approaches that only require RGB. We find that semantic segmentation performs close to using ground truth depth, whereas estimating depth values leads to a significant performance drop 15

3.1 **VLMs as a classifier:** Using video with in-context learning proves to be the strongest method, but is limited by its high latency 20

3.2 **Evaluating instruction translation:** Rephrasing the instructions causes a 37% reduction in success rate, but the agent is able to mitigate this through in-context learning. Privileged examples increase success by 17% while examples from autonomous exploration increase success by 9% 22

Acknowledgments

I would like to thank my collaborators in the Malik group, including Antonio Loquercio, Andrea Bajcsy, Himanshu Gaurav Singh, Haoran Geng who have provided me with generous guidance throughout my academic career. I'm grateful for both Jitendra Malik and Dawn Song for their oversight. I also thank Daniel Flaherty, Ana Cismaru and Tarun Amarnath for helping guide my process through graduate school and my career.

Chapter 1

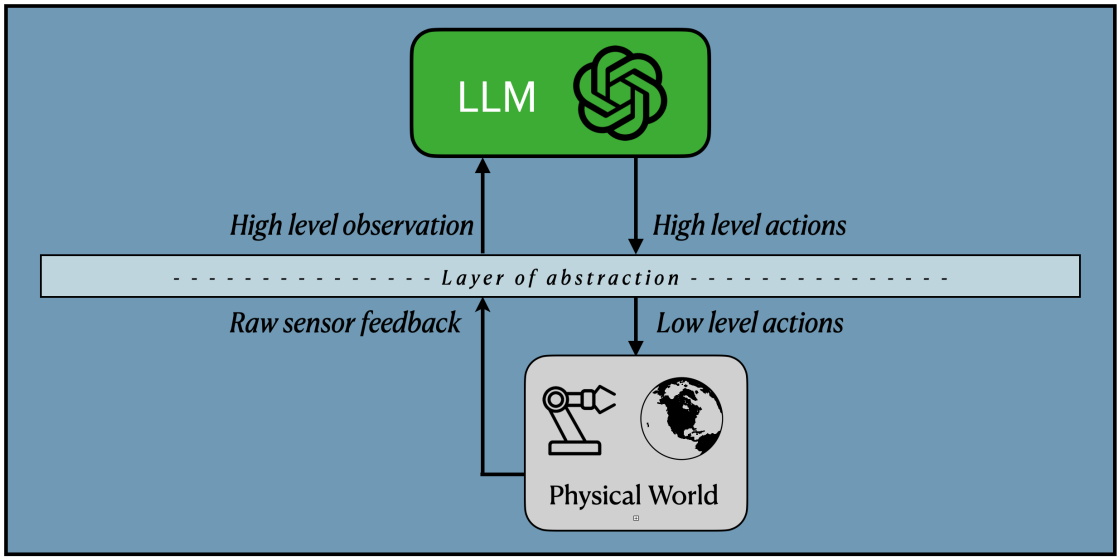
Introduction

1.1 LLMs and Robotics

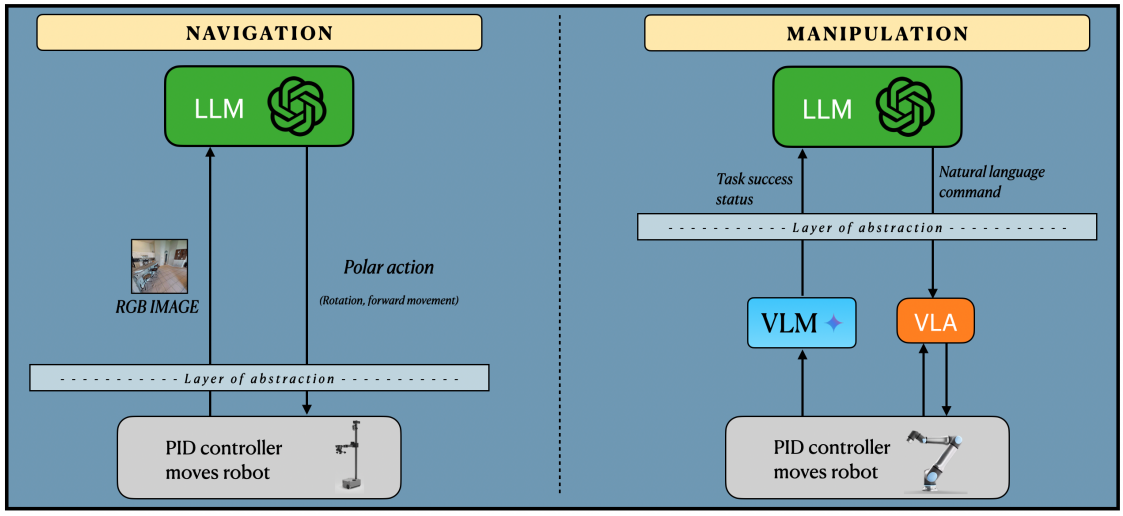
Recent years have witnessed remarkable advancements in the capabilities of Large Language Models (LLMs), as new models consistently improve state-of-the-art performance across diverse benchmarks. These models are also becoming more multi-modal, capable of reasoning across images, videos, audio, and different file types, moving well beyond plain text [46, 32]. This shift toward multi-modality moves LLMs closer to the embodied world, yet they remain isolated from any physical interaction. Nevertheless, the high-level knowledge derived from large-scale internet training holds substantial promise for achieving more generalizable robotic intelligence.

In stark contrast to these gains, robotics research has seen modest progress, particularly when it comes to generalization and scalability. While LLMs have now made their way into the lives of millions of people and significantly enhanced their everyday tasks, robotic systems have yet to achieve widespread utility or societal impact. The few commercial robot policies deployed in the real world are typically highly specialized, tailored to narrow tasks, and developed through expensive and labor-intensive data collection processes [50]. The inherent scarcity and specificity of robotics data, especially compared to the wealth of textual and visual data contained on the internet, remains a critical bottleneck for robotic advancements.

Inspired by the successful scaling trends in training data and parameter count observed with LLMs, Vision-Language-Action (VLA) models have emerged as a promising framework for developing robotic policies capable of performing diverse tasks across varying embodiments [21, 5, 6, 48]. Typically initialized with a pretrained Vision-Language Model (VLM) backbone, these models have shown some degree of internet-level knowledge [7] and language understanding. Concurrently, significant efforts have been dedicated to constructing large-scale robotic datasets comprised of clean, diverse data sources [10, 20, 15].



(a) General paradigm of an embodied LLM agent



(b) Comparing the abstraction layers for navigation and manipulation

Figure 1.1: **LLMs as embodied agents:** In our navigation framework, the LLM directly selects low-level actions based on visual input, whereas our manipulation framework delegates fine-grained actions to a specialized VLA tool, yielding a higher-level abstraction.

An alternative line of research involves connecting LLMs to external tools, allowing them to interact beyond the original text sandbox. Such frameworks have seen success in applications such as database querying [13], web navigation [22], and software engineering [18], where well-defined tools can cleanly integrate into an LLM workflow. Naturally, substantial work has been done to extend this paradigm closer to the embodied world. Recent

benchmarks ScienceWorld [52] and ALFWorld [44] have proposed simulated textual environments that mimic the real world, providing early evidence that LLMs can act in this domain. To go a step further, [1, 62, 25] enable the outputs from LLM-controlled tools to directly translate to physical actions on a robot manipulator, effectively bridging symbolic reasoning with real-world interaction. This framework offers a compelling foundation upon which this work builds, exploring new methodologies to harness the strengths of LLMs for robust, generalizable robotic intelligence.

1.2 Designing Embodied Agents

In this paper, we study methodologies to effectively connect the general intelligence of LLMs to physical robotic systems, first for navigation and then for manipulation. In both scenarios, we treat the LLM as a true agent: provided with a task description in natural language, the agent perceives its environment through structured observations and uses carefully designed tools to interact with the world. Importantly, the implementation details of these low-level tools are abstracted away from the agent, allowing it to focus on the high-level reasoning and planning aspects of the task.

Figure 1.1a shows a broad overview of our framework: the agent operates above a clearly defined *layer of abstraction*, which separates it from the physical environment. In the navigation setting, this abstraction is relatively low: the LLM directly selects discrete movement commands based on visual input (Figure 1.1b). In contrast, the manipulation setting necessitates a significantly higher layer of abstraction, due to the increased complexity of the action space. Therefore, we design the agent to issue specific language commands, which a specialized VLA module translates into fine-grained motor actions.

More concretely, Chapter 2 presents *VLMnav* [14], an embodied framework to transform a VLM into an end-to-end navigation policy. In contrast to prior work, we do not rely on a separation between perception, planning, and control; instead, we use a VLM to select actions in *one step*. Without any fine-tuning or exposure to navigation data, we show that a VLM can be used as an end-to-end policy zero-shot. This makes our approach open-ended and generalizable to any downstream navigation task. We run an extensive study to evaluate the performance of our approach in comparison to baseline prompting methods. In addition, we perform a detailed ablation study to understand the most impactful design decisions.

In Chapter 3, we study how a VLA can be leveraged as a specialized tool for an embodied LLM agent. We propose a framework to address this greater action and observation-space complexity, and analyze the design choices that let the agent most effectively leverage the VLA's capabilities.

Chapter 2

Navigation

2.1 Introduction

The ability to navigate effectively within an environment to achieve a goal is a hallmark of physical intelligence. Spatial memory, along with more advanced forms of spatial cognition, is believed to have begun evolving early in the history of land animals and advanced vertebrates, likely between 400 and 200 million years ago [28]. Because this ability has evolved over such a long period, it feels almost instinctual and trivial to humans. However, navigation is, in reality, a highly complex problem. It requires the coordination of low-level planning to avoid obstacles alongside high-level reasoning to interpret the environment's semantics and explore the directions that are most likely to get the agent to achieve their goals.

A significant portion of the navigation problem appears to involve cognitive processes similar to those required for answering long-context image and video questions, an area where contemporary VLMs excel [31, 46]. However, when naively applied to navigation tasks, these models face clear limitations. Specifically, when given a task description concatenated with an observation-action history, VLMs often struggle to produce fine-grained spatial outputs to avoid obstacles and fail to effectively utilize their long-context reasoning capabilities to support effective navigation [34, 30, 33].

To address these challenges, previous work has included VLMs as a component of a modular system to perform high-level reasoning and recognition tasks. The systems generally contain an explicit 3D mapping module and a planner to deal with the more embodied part of the task, e.g., motion and exploration [21, 27, 12, 61, 24]. While modularity has the advantage of utilizing each component only for the sub-task it excels at, it comes at the disadvantage of system complexity and task specialization.

In this chapter, we show that an off-the-shelf VLM can be used as a zero-shot and end-to-end language-conditioned navigation policy. The key idea to achieve this goal is transforming the navigation problem into something VLMs excel at: *answering a question about an image.*

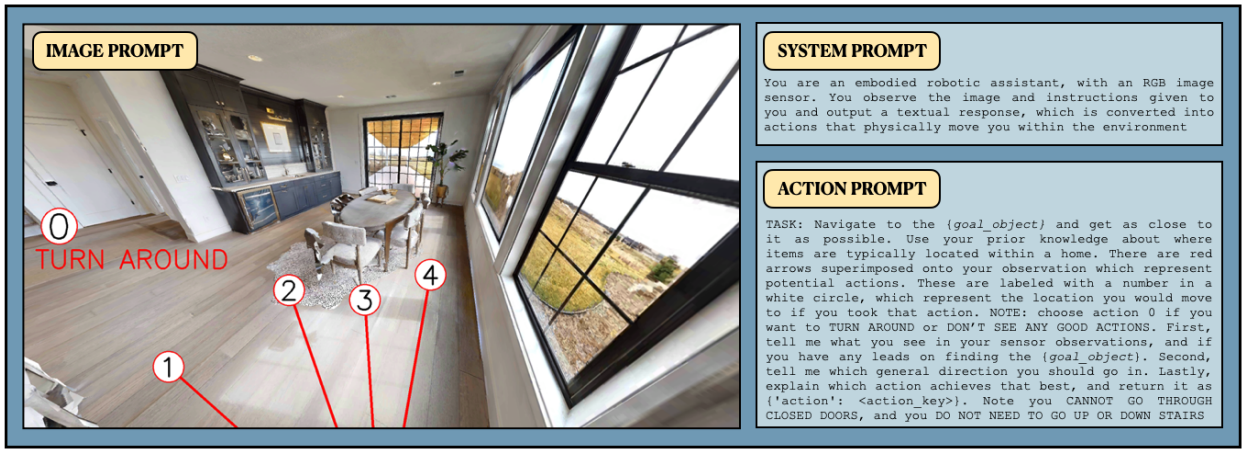


Figure 2.1: **Prompt:** The full action prompt for VLMnav consists of three parts: A system prompt to describe the embodiment, an action prompt to describe the task, the potential actions, and the output instruction, and an image prompt showing the current observation along with the annotated actions

To do so, we develop a novel prompting strategy that enables VLMs to explicitly consider the problem of exploration and obstacle avoidance. This prompting is general, in the sense that it can be used for any vision-based navigation task.

Compared to prior approaches, we do not employ modality-specific experts [35, 61, 39], do not train any domain-specific models [63, 11] and do not assume access to probabilities from the models [35, 61].

We evaluate our approach on established benchmarks for embodied navigation [57, 19], where results confirm that our method significantly improves navigation performance compared to existing prompting methods. Finally, we draw design insights from ablation experiments over several components of our embodied VLM framework.

2.2 Related Work

The most common approach for learning an end-to-end navigation policy involves training a model from scratch using offline datasets [41, 40, 9, 42, 38]. However, collecting large-scale navigation data is challenging, and as a result, these models often struggle to generalize to novel tasks or out-of-distribution environments.

An alternative approach to enhance generalization is fine-tuning existing VLMs with robot-specific data [6, 7, 21, 63]. Although this method can lead to more robust end-to-end policies, fine-tuning may destroy features not present in the fine-tuning dataset, ultimately limiting the model's generalization ability.

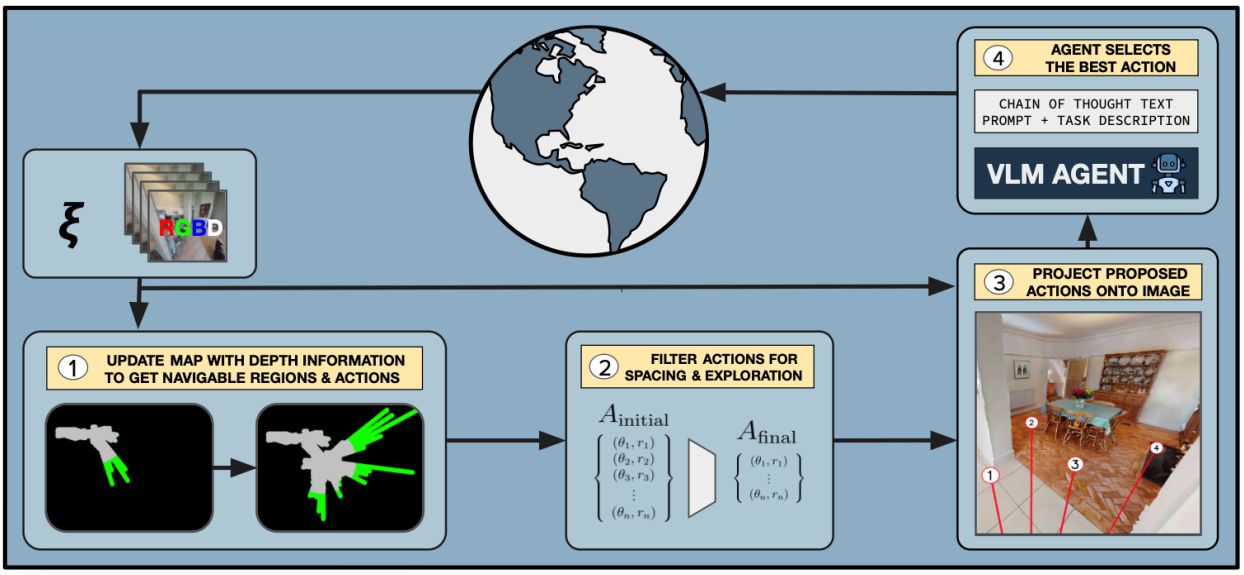


Figure 2.2: **Approach:** Our method is made up of four key components: (i) *Navigability*, which determines locations the agent can actually move to, and updates the voxel map accordingly. An example update step to the map shows the marking of new area as explored (gray) or unexplored (green). (ii) *Action Proposer*, which refines a set of final actions according to spacing and exploration. (iii) *Projection*, which visually annotates the image with actions. (iv) *Prompting*, which constructs a detailed chain-of-thought prompt to select an action.

An alternate line of work focuses on using these models zero-shot [24, 64, 61, 39, 35, 12, 30], by prompting them such that the responses align with task specifications. For instance, [12, 9] use CLIP or DETIC features to align visual observations to language goals, build a semantic map of the environment, and use traditional methods for planning. Other works design specific modules to handle the task of exploration [39, 35, 24, 49]. These systems often require an estimation of confidence to know when to stop exploring, which is commonly done using token or object probabilities [35, 61]. In addition, many of these approaches also use low-level navigation modules, which abstract away the action choices to a pre-trained point-to-point policy such as the Fast Marching Method [9, 12, 39, 24, 61].

Visual Prompting Methods: To enhance the task-specific performance of VLMs, recent work has involved physically modifying images before passing them to the VLM. Examples include [45], which annotates images to help recognize spatial concepts. [60] introduces *set-of-mark*, which assigns unique labels to objects in an image and references these labels in the textual prompt to the VLM. This visual enhancement significantly improves performance on tasks requiring visual grounding. Building on this, [22, 59] apply similar visual prompting methods to the task of web navigation and show VLMs are able to complete such tasks zero shot.

Prompting VLMs for Embodied Navigation: CoNVOI [36] overlays numerical markers

on an image and prompts the VLM to output a sequence of these markers in alignment with contextual cues (e.g., *stay on the pavement*), which is used as a navigation path. Unlike our work, they (i) rely on a low-level planner for obstacle avoidance rather than using the VLM's outputs directly as navigational actions, and (ii) do not leverage the VLM to guide the agent toward a specific goal location. PIVOT [30], introduces a visual prompting method that is most similar to ours. They approach the navigation problem by representing one-step actions as arrows pointing to labeled circles on an image. At each step, actions are sampled from an isotropic Gaussian distribution, with the mean and variance iteratively updated based on feedback from the VLM. The final action is selected after refining the distribution. While PIVOT is capable of handling various real-world navigation and manipulation tasks, it has two significant drawbacks: (i) it does not incorporate depth information to assess the feasibility of action proposals, leading to less efficient movement; and (ii) it requires many VLM calls to select a single action, resulting in higher computational costs and latency.

2.3 Overview

We present VLMnav, designed as a navigation system that takes as input goal $\mathcal{G}$, which can be specified in language or an image, RGB-D image I, pose ξ, and subsequently outputs action a. The action space consists of rotation about the yaw axis and displacement along the frontal axis in the robot frame, which allows all actions to be expressed in polar coordinates. As it is known that VLMs struggle to reason about continuous coordinates [33], we instead transform the navigation problem into the selection of an action from a discrete set of options [60]. Our core idea is to choose these action options in a way that avoids obstacle collisions and promotes exploration.

Figure 2.2 summarizes our approach. We start by determining the navigability of the local region by estimating the distance to obstacles using a depth image (Sec. 2.3). Similar to [9, 39, 35, 36, 12, 61, 49] we use the depth image and pose information to maintain a top-down voxel map of the scene, and notably mark voxels as *explored* or *unexplored*. Such a map is used by an *Action Proposer* (Sec. 2.3) to determine a set of actions that avoid obstacles and promote exploration. We then project this set of possible actions to the first-person-view RGB image with the *Projection* (Sec. 2.3) component. Finally, the VLM takes as input this image and a carefully crafted prompt, described in Sec. 2.3, to select an action, which the agent executes. To determine episode termination, we use a separate VLM call, detailed in Sec. 2.3.

Navigability

Using a depth image, we compute a *navigability mask* that contains the set of pixels that can be reached by the robot without crashing into any obstacles.

Next, for all directions $\theta \in fov$, we use the *navigability mask* to calculate the farthest straight-line distance r that the agent can travel without colliding. This creates a set of actions A_{initial} that are collision-free. Figure 2.3 illustrates an example calculation of the mask and navigable actions.

At the same time, we use the depth image and the pose information to build a 2D voxel map of the environment. All observable areas within 2 meters of the agent are marked as *explored*, and the ones beyond as *unexplored*.

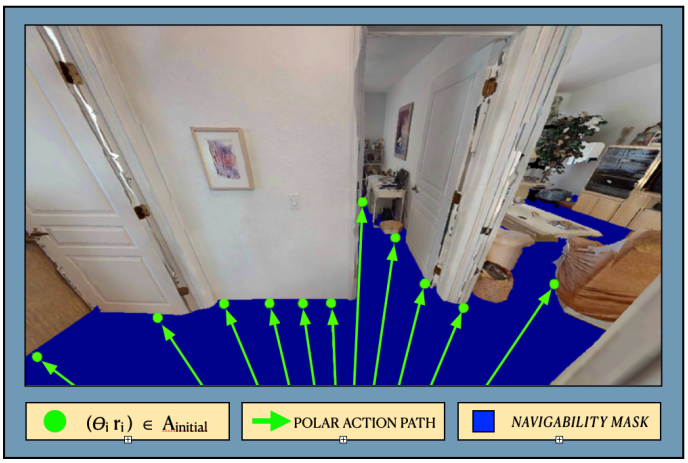


Figure 2.3: **Navigability:** An example step of the *Navigability* subroutine. The navigability mask is shown in blue and polar actions making up A_{initial} are in green

Action Proposer

We design the *Action Proposer* routine to refine $A_{\text{initial}} \rightarrow A_{\text{final}}$, an action set that is interpretable for the VLM and promotes exploration. Taking advantage of the information accumulated in our voxel map, we look at each action and define an exploration indicator variable e_i as

$$e_i = \begin{cases} 1 & \text{if region } (\theta_i, r_i) \text{ is unexplored} \\ 0 & \text{if region } (\theta_i, r_i) \text{ is explored} \end{cases}$$

To build A_{final}, we need to prioritize unexplored actions, and also ensure there is enough visual spacing between actions for the VLM to discern. We start by adding unexplored actions to A_{final} if an angular spacing of θ_δ is maintained.

$$A_{\text{final}} \leftarrow A_{\text{final}} \cup \{(\theta_i, r_i) \mid e_i = 1 \text{ and } |\theta_i - \theta_j| \geq \theta_\delta, \forall (\theta_j, r_j) \in A_{\text{final}}\}$$

To sufficiently cover all directions but still maintain an exploration bias, we supplement A_{final} by adding explored actions subject to a *larger* angular spacing of $\theta_\Delta > \theta_\delta$:

$$A_{\text{final}} \leftarrow A_{\text{final}} \cup \{(\theta_i, r_i) \mid e_i = 0 \text{ and } |\theta_i - \theta_j| \geq \theta_\Delta, \forall (\theta_j, r_j) \in A_{\text{final}}\}$$

Lastly, we want to ensure these actions don't move the agent too close to obstacles, so we clip

$$r_i \leftarrow \min(\frac{2}{3} \cdot r_i,\ r_{max}) \quad \forall (\theta_i, r_i) \in A_{\text{final}}$$

Occasionally, the agent can get stuck in a corner where there are *no* navigable actions ($A_{\text{initial}} = \emptyset$). To address this, we add a special action $(\pi, 0)$, which rotates the agent by 180°. This also allows efficient entry/exit of rooms where the agent quickly identifies that the goal is not in that room.

The proposed set A_{final} now has three important properties: (i) actions correspond to navigable paths, (ii) there is sufficient visual spacing between actions, and (iii) there is an engineered bias towards exploration. We call this approach to exploration *explore bias*.

Projection

Visually grounding these actions in a space the VLM can understand and reason about is the next step. The *Projection* component takes in A_{final} from 2.3 and RGB image I, and outputs annotated image $\hat{I}$. Similarly to [30], each action is assigned a number and overlayed onto the image. We assign the special rotation action with 0 and annotate it onto the side of the image along with a label *Turn Around*. We find that visually annotating it, instead of just describing it in the textual prompt, helps ground its probability of being chosen to that of the other actions.

Prompting

To elicit a final action, we craft a detailed textual prompt T, which is fed into the VLM along with $\hat{I}$. This prompt primarily describes the details of the task, the navigation goal, and how to interpret the visual annotations. Additionally, we ask the model to describe the spatial layout of the image and to make a high-level plan *before* choosing the action, which serves to improve reasoning quality as found by [55, 23]. For image-based navigation goals, the goal image is simply passed into the VLM in addition to T and $\hat{I}$. The full prompt can be found in Figure 2.1.

The action chosen by the VLM, $P_{\text{vlm}}(a^*|\hat{I}, T) \in A_{\text{final}}$ is then directly executed in the environment. Notably, this does not involve any low-level obstacle avoidance policy as in other works [9, 39, 12, 61, 24].

Termination

To complete a navigation task, the agent must terminate the episode by calling special action *stop* within a threshold distance of the goal object. Compared to other approaches that leverage a low-level navigation policy [9, 39, 12, 61, 24], our method does not explicitly choose a target coordinate location to navigate to, and therefore we face an additional challenge of determining when to stop. Our solution is to use a separate VLM prompt that

TERMINATION PROMPT

```
The agent has the following navigation task: \n{task}\n. The agent
has sent you an image taken from its current location. Your job is to
determine whether the agent is close to the specified {goal_object}.
First, tell me what you see in the image, and tell me if there is a
{goal_object} that matches the description. Then, return 1 if the
agent is close to the {goal_object}, and 0 if it isn't. Format your
answer in the json {'done': <1 or 0>}
```

Figure 2.4: **Prompting:** The separate prompt for determining episode termination

explicitly asks whether or not to stop, which is shown in Figure 2.4. We do this for two reasons:

1. Annotations: The arrows and circles from Sec. 2.3 introduce noise and clutter to the image, making it more difficult to understand.
2. Separation of tasks. To avoid any task interference, the action call is only concerned with navigating and the stopping call is only concerned with stopping.

To avoid terminating the episode too far away from the object, we terminate the episode when the VLM calls *stop* two times in a row. After the VLM calls *stop* the first time, we turn off the navigability and explore bias components to ensure the agent doesn't move away from the goal object.

2.4 Experiments

We evaluate our approach on two popular embodied navigation benchmarks, ObjectNav [3] and GoatBench [19], which use scenes from the Habitat-Matterport 3D dataset [58, 37]. Further, we analyze how the performance of an end-to-end VLM agent changes with variations in design parameters such as field-of-view, length of the contextual history used to prompt the model, and quality of depth perception.

Setup: Similar to [57], the agent adopts a cylindrical body of radius 0.17m and height 1.5m. We equip the agent with an egocentric RGB-D sensor with resolution (1080, 1920) and a horizontal field-of-view (FOV) of 131°. The camera is tilted down with a pitch of 25° similar to [35], which helps determine navigability. We use Gemini Flash as the VLM for all our experiments, given its low cost and high effectiveness.

Metrics: As in prior work [19, 57, 2], we use the following metrics: (i) Success Rate (SR): fraction episodes that are successfully completed (ii) Success Rate Weighted by Inverse Path Length (SPL): a measure of path efficiency.

Baselines: We use PIVOT [30] as a baseline as it is most similar to ours. To investigate the impact of our action selection method, we ablate it by evaluating *Ours w/o nav*: the same as ours but without the *Navigability* and *Action Proposer* components. The action choices for this baseline are a static set of evenly-spaced action choices, including the *turn around* action. Notably, these actions do not consider navigability or exploration. To further evaluate the impact of visual annotation, we also evaluate a baseline *Prompt Only*, which sees actions described in text ("turn around", "turn right", "move forward", ...) but not annotated visually. These different prompting baselines can be visualized in Fig 2.5.

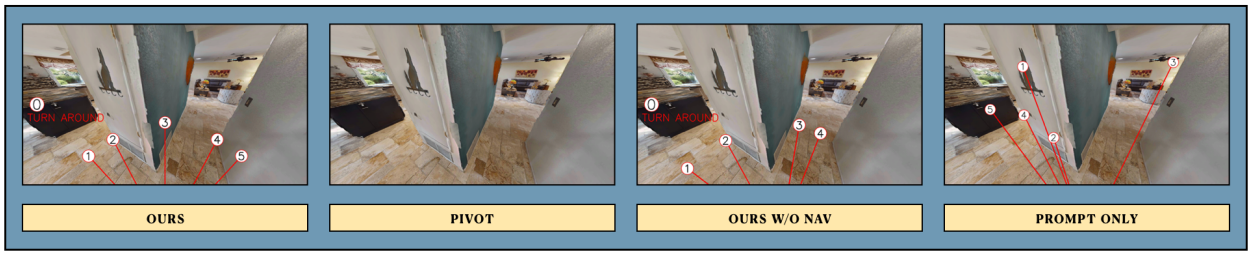


Figure 2.5: **Baselines:** Comparing the four different methods on a sample image. *Ours* contains arrows that point to navigable locations, *PIVOT* has arrows sampled from a random 2-D Gaussian, *Ours w/o nav* sees uniformly spaced arrows (note arrows 3 and 5 point into a wall), and *Prompt Only* sees just the raw RGB image

We note that in our experiments and baselines, we turn the *allow_slide* parameter on, which allows the agent to slide against obstacles in the simulator. Our experiments show that removing this assumption leads to large drops in performance.

ObjectNav

The Habitat ObjectNav benchmark requires navigation to an object instance from one of six categories *[Sofa, Toilet, TV, Plant, Chair, Bed]*. As in [57], to get the optimal path length, we take the minimum of the shortest paths to all instances of the object. These experiments are evaluated with a success threshold of 1.2 meters [39].

Table 2.1 summarizes our results. Our method outperforms PIVOT by over 25%, and nearly doubles its navigation efficiency in terms of SPL. We see that our action selection method is highly effective as shows a 17% improvement over *Ours w/o nav*. Removing visual annotations leads to a slight decrease in success rate but a significant reduction in SPL, indicating that visual grounding is important for navigation efficiency. Interestingly, we find that PIVOT performs worse than both of our ablations. We attribute this to limited expressivity

Run	SR	SPL
Ours	**50.4%**	**0.210**
Ours w/o nav	33.2%	0.136
Prompt Only	29.8%	0.107
PIVOT [30]	24.6%	0.106
Ours w/o sliding	12.9%	0.063

Table 2.1: **ObjectNav Results:** We evaluate four different prompting strategies on the ObjectNav benchmark, and see our method achieves highest performance in both accuracy (SR) and efficiency (SPL). Ablating the *allow_slide* parameter shows our method is dependent on sliding past obstacles

in its action space, which prevents it from executing large rotations or turning around fully. This often leads to the agent getting stuck in corners, hindering its ability to recover and navigate effectively.

We note that disabling sliding results in a large drop in performance, signaling that while effective in simulation, our method would likely lead to collisions with obstacles in the real world. While our *Navigability* module can identify navigable locations, it does not consider the specific size and shape of the robot in this calculation, leading to occasional collisions where the agent gets stuck since we lack an explicit action to backtrack previous motions.

Go To Anything Benchmark (GOAT)

GOAT Bench [19] is a recent benchmark that establishes a higher level of navigation difficulty. Each episode contains 5-10 sub-tasks across three different goal modalities: (i) Object names, such as *refrigerator*, (ii) Object images, and (iii) Detailed text descriptions such as *Grey couch located on the left side of the room, next to the picture and the pillow.* Table 2.2 shows our results, evaluated on the val unseen split.

Run	SR	SPL	Image SR	Object SR	Description SR
Ours	**16.3%**	**0.066**	**14.3%**	**20.5%**	**13.4%**
Ours w/o nav	11.8%	0.054	7.8%	16.5%	10.2%
Prompt Only	11.3%	0.037	7.7%	15.6%	10.1%
PIVOT [30]	8.3%	0.038	7.0%	11.3%	5.9%

Table 2.2: **GOAT Results:** Comparison of prompting strategies on GOAT Bench, a more challenging navigation task. Across three different goal modalities, our method strongly outperforms baseline methods

Across all goal modalities, our model achieves significant improvements over baselines. These improvements are especially evident in image goals, where our model achieves nearly twice the success rate of all baseline methods. This highlights the robustness and general nature of our system. As with the ObjectNav results, *Ours w/o nav* and *Prompt only* perform comparable, and both outperform PIVOT. For all prompting methods, the image and description

modalities prove more challenging than the object modality, similarly to what was found by [19].

Comparison to state-of-the-art: We turn the *allow_slide* parameter off and compare to two state-of-the-art specialized approaches: (i) SenseAct-NN [19] is a policy trained with reinforcement learning, using learned submodules for different skills; and (ii) Modular GOAT [9] is a compound system that builds a semantic memory map of the environment and uses a low-level policy to navigate to objects within this map. Unlike SenseAct-NN, our work is zero-shot, and unlike Modular GOAT, we do not rely on a low-level policy or a separate object-detection module.

Run	**SR**	**SPL**
SenseAct-NN Skill Chain	**29.5%**	0.113
Modular GOAT	24.9%	**0.172**
Ours w/ sliding	16.3%	0.066
Ours	6.9%	0.049

Table 2.3: **GOAT comparison:** Directly comparing to other works, we see that specialized systems still produce superior performance. We also note these other works use a narrower FOV, lower image resolution, and a different action space, which could explain some of the differences

We compare the results of our approach to these baselines in Table 2.3. Interestingly, these methods have different strengths: a reinforcement learning approach leads to the highest success rate. Conversely, the modular navigation system achieves the highest navigation efficiency.

Our method shows lower performance compared to these specialized baselines across both metrics, even when permitted to slide over obstacles. Notably, we observe that in 13.9% of the runs, the VLM prematurely calls *stop* when it is between 1 to 1.5 meters from the target object. These instances are classified as failures, as the benchmark defines a run as successful only if the agent is within 1 meter of the object. This finding suggests that our VLM lacks the fine-grained spatial awareness necessary to accurately assess distances to objects. However, it also indicates that in over 30% of the runs, our VLM agent is able to approach the goal object closely, highlighting its capability to reach near-target positions.

As shown in previous experiments, when not allowed to slide over objects, our approach's performance drastically decreases, as it gets frequently blocked between obstacles and does not have a way to backtrack its actions.

Exploring the design space of VLM agents for navigation

In this section, we look at major design choices that impact the navigation ability of VLM-based agents in our setup, all evaluated on the ObjectNav dataset.

How important is camera FOV for navigation?

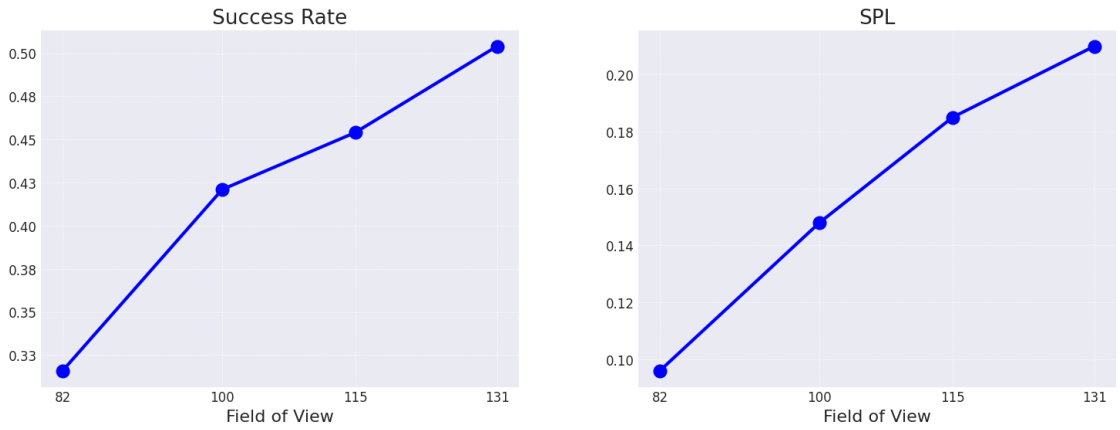


Figure 2.6: **Impact of sensor FOV:** We evaluate the performance of four different sensor FOVs, and find that a wider FOV invariably leads to higher performance

An agent's navigation abilities largely depend on how fine-grained its vision is. In this section, we study whether our VLM agent can benefit from high-resolution images. Specifically, we run our method using four different FOVs: 82° [57], 100°, 115° and 131° (iPhone 0.5 camera). The results of this experiment, shown in Fig. 2.6, indicate positive scaling behaviors on both navigation accuracy and efficiency.

Do longer observation-action histories help?

In this section, we study whether a VLM navigation agent can effectively use a history of observations. We create a prompt containing the observation history in a naive way, i.e., we concatenate observations and actions from the K most recent environment steps and feed this into the VLM as context. For all these experiments, we remove our exploration bias (see Sec. 2.3) to specifically isolate the contribution of a longer history.

History Length	SR	SPL
No history	46.8%	0.193
5	42.7%	0.180
10	45.4%	0.196
15	40.4%	0.170

Table 2.4: **Impact of adding context history:** We compare our method to alternatives of keeping the past 0, 5, 10, and 15 observations and actions. We see that adding context history does not improve the performance of our method

The results of these experiments are shown in Table 2.4. We find that when naively concatenating past observations and actions, our prompt strategy is unable to use a longer context. Indeed, the performance remains the same or decreases when increasing the history length.

How important is perfect depth perception?

Within the simulator, the depth sensor provides accurate pixel-wise depth information, which is important for determining the navigability mask. To investigate the importance of quasi-perfect depth perception, we evaluate two alternate approaches that only use RGB: (i) **Segformer**, which uses [56] to semantically segment pixels belonging to the *floor* region. We use this region as the *navigability mask* and bypass the need for any depth information. We estimate the distances to obstacles by multiplying the number of pixels with a constant factor. (ii) **ZoeDepth**, which uses [4] to estimate metric depth values. We use such predicted values instead of the ground-truth distances from the simulator and compute navigability in the original way.

Run	SR	SPL
Depth sensor	50.4%	0.210
Segformer [56]	47.2%	0.183
ZoeDepth [4]	39.1%	0.161

Table 2.5: **Depth Ablation:** We evaluate two alternate approaches that only require RGB. We find that semantic segmentation performs close to using ground truth depth, whereas estimating depth values leads to a significant performance drop

The results of this study are presented in Table 2.5. We find that depth estimation from [4] is not accurate enough to identify navigable areas. Indeed, depth noise leads to a 10% drop in SR. However, using a segmentation mask instead of relying on depth information surprisingly proves to be quite effective, with only a decrease of 3% with respect to using perfect depth perception. Overall, our experiments show that a VLM-based navigation agent can perform well with only RGB information.

2.5 Conclusion

In this chapter, we present VLMnav, a novel visual prompt-engineering approach that enables an off-the-shelf VLM to act as an end-to-end navigation policy. The main idea behind this approach is to carefully select action proposals and project them on an image, effectively transforming the problem of navigation into one of question-answering. Through evaluations on the ObjectNav and GOAT benchmarks, we see significant performance gains over the iterative baseline PIVOT, which was the previous state-of-the-art in prompt engineering for visual navigation. Our design study further highlights the importance of a wide field of view and the possibility of deploying our approach with minimal sensing, i.e., only an RGB image.

Our method has a few limitations. The drastic decrease in performance from disabling the *allow_slide* parameter indicates that there are several collisions with obstacles, which could be problematic in a real-world deployment. In addition, we find that specialized systems such

as [19] outperform our work. However, as the capabilities of VLMs continue to improve, we hypothesize that our approach could help future VLMs reach or surpass the performance of specialized systems for embodied tasks.

Chapter 3

Manipulation

3.1 Introduction

Vision-Language-Action (VLA) models [21, 7, 48, 5] have exhibited early promise in performing diverse tasks through natural language prompting. However, as highlighted in [54], current VLA models have difficulty interpreting and executing complex tasks. To address these shortcomings, this work proposes a framework that leverages the strengths of LLMs as central planning agents in conjunction with VLA models. The core idea is to have the planning agent interact with the environment by prompting a VLA, observing the outcome of the robotic trajectory executed by the VLA, and iteratively refining the subsequent instructions. In addition, this work makes the following contributions. First, we evaluate several approaches for autonomously communicating feedback and task status from the robot environment to the planning agent. Second, we present a sensitivity analysis of an off-the-shelf VLA to variations in language prompting. Third, we show that through exploration, the planning agent can autonomously improve its ability to use the VLA.

3.2 Related Work

Connecting the rich space of language commands with the world of robotic policies has been attempted in a few ways. As mentioned previously, VLAs [21, 7, 48, 5, 47], are one approach to developing generalist policies that span natural language inputs. A survey paper [54], extensively evaluates several VLAs and concludes that OpenVLA [21] achieves the highest performance, but like the other models, is very sensitive to out-of-distribution inputs.

Another line of existing research leverages LLMs as high-level planners that call low-level robotic primitives [1, 51, 25, 62, 16], which has enabled significant improvements in the range of language commands a robot can execute. However, these low-level primitives are often limited in scope and constrain the robot to a set of pre-defined behaviors.

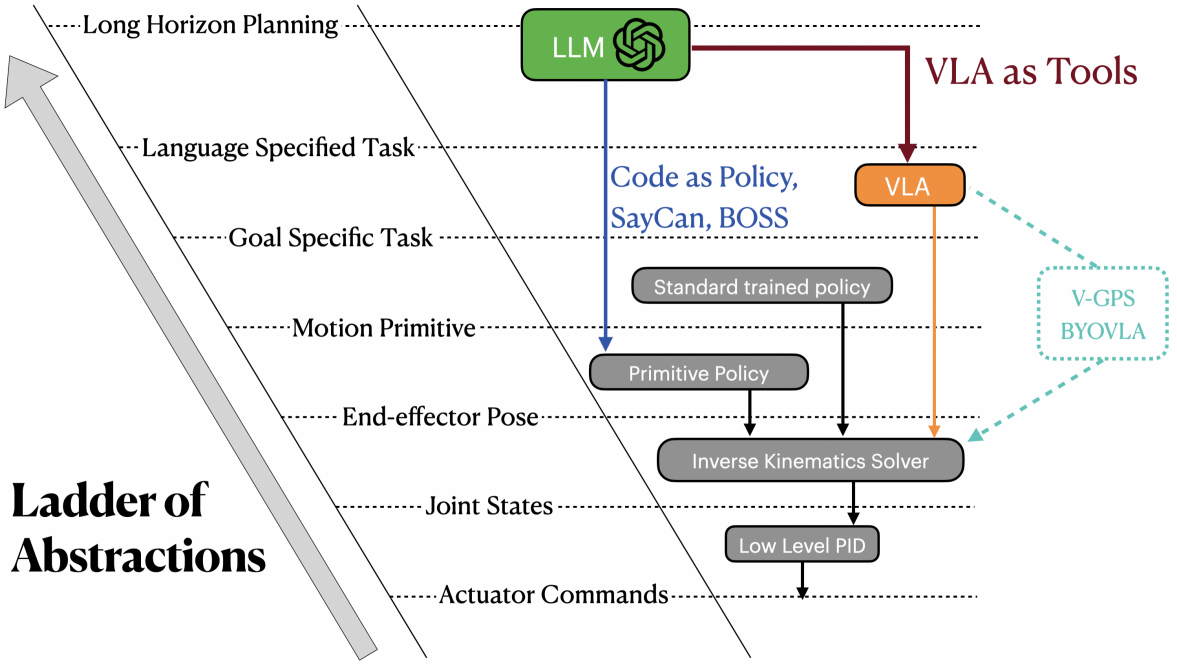


Figure 3.1: **Robotic generalists require several layers of abstraction:**. VLAs can generalize across tasks, but they fall short of the high-level planning capabilities of LLMs. Works shown in blue bridge this gap by allowing LLMs to solve tasks by calling low-level robotic APIs. This work builds off of those ideas by connecting LLMs to VLAs, enabling a higher level of abstraction

VLMs have also gained traction as a method for autonomously evaluating robotic performance due to their improved visual capabilities [65, 53, 26]. This has taken the form of training reward functions, estimating task completion values, and classifying robotic trajectories.

Additionally, some recent works explore ways in which to improve VLAs in a zero-shot setting, treating them as black-box models [17, 29, 43]. Surprisingly, an area that has been less explored is employing LLMs as planners to use VLAs as black-box tools, taking advantage of the shared language space between both models.

Figure 3.1 illustrates the many levels of abstractions present in the manipulation space, and where previous works fall. Highlighted in red, our approach builds a direct connection between high-level LLM reasoning and low-level execution by prompting a VLA as an intermediate policy layer.

3.3 VLA as Tools

The core idea of the proposed system involves a central planning agent that can send instructions for a VLA to execute, effectively making the VLA a *tool*. Unlike the traditional agent and tool paradigm, which has seen tremendous success in tasks such as web navigation and software development, the problem setting here poses two significant challenges: (i) the tool icannot be documented and described easily, and (ii) the time horizon at which actions are executed is also unknown. To address the former, we take advantage of the ability for LLMs to learn by example [8, 55]. To address the latter, we design a feedback module to periodically provide the agent with updates on the impact of its previous actions.

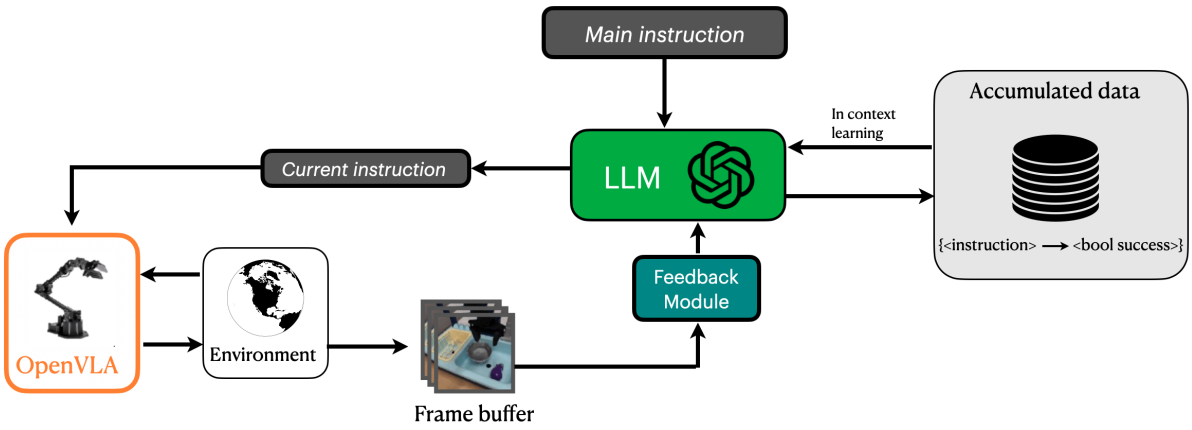


Figure 3.2: **Overview of the proposed framework:** In one process, the VLA continuously executes an instruction in its environment. Concurrently, a central planning agent chooses what instructions to send to the VLA, and a feedback module periodically sends updates to the planning agent. The trajectory results are stored offline for future prompting of the central agent

A system overview can be seen in Figure 3.2. The VLA operates at high frequency, executing some instruction provided by the planning agent, and writing third-person images to a shared memory buffer. The feedback module periodically reads from this buffer and reports whether the current instruction was completed successfully, failed, or is still in progress. The central planning agent (green), upon receiving this feedback, decides what subsequent instruction to send to the VLA. The individual results of the attempted tasks are then stored offline for future prompting of the planning agent.

In addition to the proposed system, this work investigates three hypotheses.

1. LLMs can visually determine the completion status of a task
2. The space of instructions that VLAs can successfully execute is limited
3. LLMs can learn to translate general instructions into such a space

Setup: We utilize OpenVLA as the VLA model, given that it is the best available open-source model [54]. As in [21], we use the Libero environment and task suites, which contain a diverse selection of manipulation tasks. The environment provides rendered third-person images, which are used as input to OpenVLA, and also provides ground truth success information for each task. We use Gemini-flash-1.5[46] for both the central planning agent and the feedback module.

What is the best design for a feedback module?

VLMs have gained popularity in robotics in a variety of ways. [53] use VLM feedback to train a reward function for various robotic tasks. [26] propose a method for VLMs to estimate a task completion percentage from a sequence of frames. Most similar to this work, [65] use a VLM to classify the success value of a trajectory from the last frame. Surprisingly, there has been little work on passing robotic videos into the VLM as opposed to just single images. To address this, we present a small-scale study on four different designs: using just the last frame as in [65], using a video of the whole trajectory, and both methods but with a prompt consisting of several example trajectories and their respective labels, labeled as ICL[8]. The VLM is then asked to output a binary success value for each robot trajectory, and this is compared to the ground-truth success value to calculate accuracy.

Method	**Accuracy**	**Accuracy w ICL**	**Latency**
Last frame	56%	61% (+5%)	**1.5s**
Video	**60%**	**78% (+18%)**	6s

Table 3.1: **VLMs as a classifier:** Using video with in-context learning proves to be the strongest method, but is limited by its high latency

As seen in Table 3.1, VLMs are indeed able to reason about robotic videos, likely due to recent advances in context length and vision capabilities. Videos also capture important information that may not be visible in the last frame. Surprisingly, adding few-shot examples improves the video approach by a much larger margin, which is contrary to Gemini's documentation that recommends only using one video per prompt. However, the video method comes with a significant efficiency drawback, as most of the additional latency comes from sending the video over the network. From these results, we choose to use a video-based feedback module that uses in-context learning.

How sensitive is OpenVLA to variations in prompting language?

To investigate the space of instructions OpenVLA is able to successfully complete, we use an LLM to generate a new dataset of tasks, where each original task is mapped to several slightly rephrased tasks. For example, the original task *put the wine bottle on the rack* might get rephrased to *locate the wine bottle and transfer it to the wine rack*.

Figure 3.3 shows how the success rate on the rephrased instructions compares to the original success rate. Evidently, these small changes in the instruction language lead to very large decreases in performance, especially on the harder task suites *goal* and *long*. While these rephrased instructions are out of the model's training distribution, the performance drop is still concerning, as we expect the model's pre-trained language encoder to be able to handle these instructions.

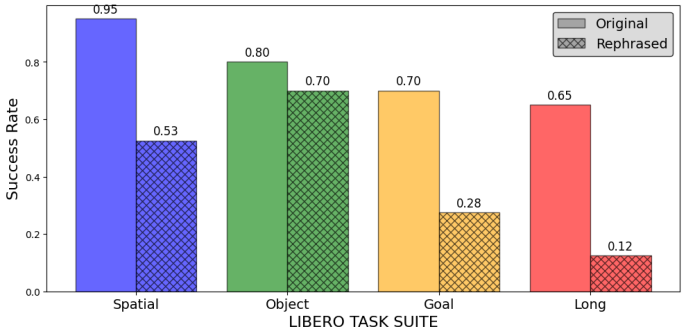


Figure 3.3: **Language sensitivity analysis:** OpenVLA is very sensitive to the specific phrasing used in the instruction

Can a planning agent learn to use a VLA autonomously?

With the findings from Sec 3.3, the natural question to ask is whether the planning agent can learn to translate these instructions into the space that the VLA can successfully complete. To investigate this, we design a prompt that includes several instruction-success tuples sampled from the other task suites, which were collected in the previous experiment. The LLM then reads the current (rephrased) instruction, along with these sampled results, and then outputs a modified instruction for the VLA to execute. With this method, the average success rate across the task suites **increases by 17%**, indicating that the planning agent can learn the space of successful VLA instructions to some degree.

However, this study relies on ground-truth success information from the simulator, which is not practical for real-world deployment, and furthermore, the in-context examples were not chosen by the agent. To investigate whether the previous results can be reproduced without any supervision or privileged information, we propose the following exploration procedure: First, the planning agent is given an image of the scene and is told to learn the capabilities of the robot. It then outputs a list of tasks for the VLA to execute, importantly involving no knowledge of what tasks the VLA was trained on. The VLA then attempts each proposed task, and a success value is derived from the VLM feedback module, instead of

the environment. These instruction-success tuples are then saved and sampled in the same prompting mechanism.

Method	Success Rate
Original	77.5%
Rephrased	40.6%
Rephrased ICL	57.5%
Exploration ICL	49.4%

Table 3.2: **Evaluating instruction translation:** Rephrasing the instructions causes a 37% reduction in success rate, but the agent is able to mitigate this through in-context learning. Privileged examples increase success by 17% while examples from autonomous exploration increase success by 9%

As seen in Table 3.2, this exploration method leads to a **9% increase** in success rate. While not as strong of an increase as using the ground-truth data, this result shows a promise of autonomous learning. We believe this method to be most limited by the VLM feedback signal, as we observe it to be unreliable at times, especially when the agent proposes strange tasks.

Conclusion and Limitations

In this chapter, we present an exploratory framework aimed at bridging the fine-grained control capabilities of VLA models with the robust planning abilities of LLMs. Recognizing the limitations of current VLAs, particularly their sensitivity to instruction phrasing and constrained execution capabilities, we introduce an LLM-based planning agent that dynamically leverages VLAs as execution tools.

Our evaluation revealed several key insights: VLM-based feedback modules, especially when leveraging video inputs and few-shot prompting, are mostly able to assess the accuracy of robotic tasks. Furthermore, we found significant sensitivity in OpenVLA's performance with respect to subtle variations in prompt phrasing, highlighting a weakness that LLM agents are well-positioned posed to address.

Encouragingly, the our agent demonstrated a ability to learn effective translations from general instructions into actionable commands for the VLA, achieving performance gains even without direct access to ground-truth task success information. However, exploration method, while promising, faced limitations primarily due to inaccuracies in the VLM's feedback signal, especially when confronted with unconventional tasks.

We acknowledge that our experiments were conducted in a purely simulated setting, which differs significantly from the real-world image distributions on which OpenVLA was originally

trained. This discrepancy likely constrained the VLA's execution effectiveness, regardless of the planning agent's proficiency. We believe deploying similar LLM-VLA systems to real world robots presents an exciting direction of future work. Despite these challenges, this exploratory work lays a path toward integrating high-level planning and generalized robotic execution, establishing a preliminary method for future advancements in autonomous manipulation systems.

Bibliography

[1] Michael Ahn et al. *Do As I Can, Not As I Say: Grounding Language in Robotic Affordances*. 2022. arXiv: 2204.01691 [cs.RO]. URL: https://arxiv.org/abs/2204.01691.

[2] Peter Anderson et al. *On Evaluation of Embodied Navigation Agents*. 2018. arXiv: 1807.06757 [cs.AI]. URL: https://arxiv.org/abs/1807.06757.

[3] Dhruv Batra et al. *ObjectNav Revisited: On Evaluation of Embodied Agents Navigating to Objects*. 2020. arXiv: 2006.13171 [cs.CV]. URL: https://arxiv.org/abs/2006.13171.

[4] Shariq Farooq Bhat et al. *ZoeDepth: Zero-shot Transfer by Combining Relative and Metric Depth*. 2023. arXiv: 2302.12288 [cs.CV]. URL: https://arxiv.org/abs/2302.12288.

[5] Kevin Black et al. *π_0: A Vision-Language-Action Flow Model for General Robot Control*. 2024. arXiv: 2410.24164 [cs.LG]. URL: https://arxiv.org/abs/2410.24164.

[6] Anthony Brohan et al. "Rt-1: Robotics transformer for real-world control at scale". In: *arXiv preprint arXiv:2212.06817* (2022).

[7] Anthony Brohan et al. "Rt-2: Vision-language-action models transfer web knowledge to robotic control". In: *arXiv preprint arXiv:2307.15818* (2023).

[8] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[9] Matthew Chang et al. "Goat: Go to any thing". In: *arXiv preprint arXiv:2311.06430* (2023).

[10] Open X-Embodiment Collaboration et al. *Open X-Embodiment: Robotic Learning Datasets and RT-X Models*. https://arxiv.org/abs/2310.08864. 2023.

[11] Kiana Ehsani et al. "Imitating Shortest Paths in Simulation Enables Effective Navigation and Manipulation in the Real World". In: *arXiv* (2023). eprint: 2312.02976.

[12] Samir Yitzhak Gadre et al. "Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2023, pp. 23171–23181.

[13] Dawei Gao et al. *Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation*. 2023. arXiv: 2308.15363 [cs.DB]. URL: https://arxiv.org/abs/2308.15363.

[14] Dylan Goetting, Himanshu Gaurav Singh, and Antonio Loquercio. *End-to-End Navigation with Vision Language Models: Transforming Spatial Reasoning into Question-Answering*. 2024. arXiv: 2411.05755 [cs.RO]. URL: https://arxiv.org/abs/2411.05755.

[15] Kristen Grauman et al. *Ego4D: Around the World in 3,000 Hours of Egocentric Video*. 2022. arXiv: 2110.07058 [cs.CV]. URL: https://arxiv.org/abs/2110.07058.

[16] Huy Ha, Pete Florence, and Shuran Song. *Scaling Up and Distilling Down: Language-Guided Robot Skill Acquisition*. 2023. arXiv: 2307.14535 [cs.RO]. URL: https://arxiv.org/abs/2307.14535.

[17] Asher J. Hancock, Allen Z. Ren, and Anirudha Majumdar. *Run-time Observation Interventions Make Vision-Language-Action Models More Visually Robust*. 2024. arXiv: 2410.01971 [cs.RO]. URL: https://arxiv.org/abs/2410.01971.

[18] Carlos E. Jimenez et al. *SWE-bench: Can Language Models Resolve Real-World GitHub Issues?* 2024. arXiv: 2310.06770 [cs.CL]. URL: https://arxiv.org/abs/2310.06770.

[19] Mukul Khanna* et al. *GOAT-Bench: A Benchmark for Multi-Modal Lifelong Navigation*. 2024. arXiv: 2404.06609 [cs.AI].

[20] Alexander Khazatsky et al. *DROID: A Large-Scale In-The-Wild Robot Manipulation Dataset*. 2024. arXiv: 2403.12945 [cs.RO]. URL: https://arxiv.org/abs/2403.12945.

[21] Moo Jin Kim et al. "OpenVLA: An Open-Source Vision-Language-Action Model". In: *arXiv preprint arXiv:2406.09246* (2024).

[22] Jing Yu Koh et al. *VisualWebArena: Evaluating Multimodal Agents on Realistic Visual Web Tasks*. 2024. arXiv: 2401.13649 [cs.LG]. URL: https://arxiv.org/abs/2401.13649.

[23] Takeshi Kojima et al. "Large language models are zero-shot reasoners". In: *Advances in neural information processing systems* 35 (2022), pp. 22199–22213.

[24] Yuxuan Kuang, Hai Lin, and Meng Jiang. *OpenFMNav: Towards Open-Set Zero-Shot Object Navigation via Vision-Language Foundation Models*. 2024. arXiv: 2402.10670 [cs.CL]. URL: https://arxiv.org/abs/2402.10670.

[25] Jacky Liang et al. *Code as Policies: Language Model Programs for Embodied Control*. 2023. arXiv: 2209.07753 [cs.RO]. URL: https://arxiv.org/abs/2209.07753.

[26] Yecheng Jason Ma et al. *Vision Language Models are In-Context Value Learners*. 2024. arXiv: 2411.04549 [cs.RO]. URL: https://arxiv.org/abs/2411.04549.

[27] Arjun Majumdar et al. "Zson: Zero-shot object-goal navigation using multimodal goal embeddings". In: *Advances in Neural Information Processing Systems* 35 (2022), pp. 32340–32352.

[28] Rubén N. Muzio and Verner P. Bingman. "Brain and Spatial Cognition in Amphibians: Stem Adaptations in the Evolution of Tetrapod Cognition". In: *Evolution of Learning and Memory Mechanisms*. Cambridge University Press, 2022, pp. 105–124.

[29] Mitsuhiko Nakamoto et al. *Steering Your Generalists: Improving Robotic Foundation Models via Value Guidance*. 2024. arXiv: 2410.13816 [cs.RO]. URL: https://arxiv.org/abs/2410.13816.

[30] Soroush Nasiriany et al. *PIVOT: Iterative Visual Prompting Elicits Actionable Knowledge for VLMs*. 2024. arXiv: 2402.07872 [cs.RO].

[31] OpenAI et al. *GPT-4 Technical Report*. 2024. arXiv: 2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

[32] OpenAI et al. *GPT-4o System Card*. 2024. arXiv: 2410.21276 [cs.CL]. URL: https://arxiv.org/abs/2410.21276.

[33] Pooyan Rahmanzadehgervi et al. *Vision language models are blind*. 2024. arXiv: 2407.06581 [cs.AI]. URL: https://arxiv.org/abs/2407.06581.

[34] Santhosh Kumar Ramakrishnan et al. *Does Spatial Cognition Emerge in Frontier Models?* 2024. arXiv: 2410.06468 [cs.AI]. URL: https://arxiv.org/abs/2410.06468.

[35] Allen Z. Ren et al. "Explore until Confident: Efficient Exploration for Embodied Question Answering". In: *arXiv preprint arXiv:2403.15941*. 2024.

[36] Adarsh Jagan Sathyamoorthy et al. *CoNVOI: Context-aware Navigation using Vision Language Models in Outdoor and Indoor Environments*. 2024. arXiv: 2403.15637 [cs.RO]. URL: https://arxiv.org/abs/2403.15637.

[37] Manolis Savva et al. *Habitat: A Platform for Embodied AI Research*. 2019. arXiv: 1904.01201 [cs.CV]. URL: https://arxiv.org/abs/1904.01201.

[38] Dhruv Shah et al. *GNM: A General Navigation Model to Drive Any Robot*. 2023. arXiv: 2210.03370 [cs.RO]. URL: https://arxiv.org/abs/2210.03370.

[39] Dhruv Shah et al. "Navigation with Large Language Models: Semantic Guesswork as a Heuristic for Planning". In: *7th Annual Conference on Robot Learning*. 2023. URL: https://openreview.net/forum?id=PsV65r0itpo.

[40] Dhruv Shah et al. *Rapid Exploration for Open-World Navigation with Latent Goal Models*. 2023. arXiv: 2104.05859 [cs.RO]. URL: https://arxiv.org/abs/2104.05859.

[41] Dhruv Shah et al. "ViNG: Learning Open-World Navigation with Visual Goals". In: *2021 IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 13215–13222. DOI: 10.1109/ICRA48506.2021.9561936.

[42] Dhruv Shah et al. *ViNT: A Foundation Model for Visual Navigation*. 2023. arXiv: 2306.14846 [cs.RO]. URL: https://arxiv.org/abs/2306.14846.

[43] Lucy Xiaoyang Shi et al. *Hi Robot: Open-Ended Instruction Following with Hierarchical Vision-Language-Action Models*. 2025. arXiv: 2502.19417 [cs.RO]. URL: https://arxiv.org/abs/2502.19417.

[44] Mohit Shridhar et al. *ALFWorld: Aligning Text and Embodied Environments for Interactive Learning*. 2021. arXiv: 2010.03768 [cs.CL]. URL: https://arxiv.org/abs/2010.03768.

[45] Aleksandar Shtedritski, Christian Rupprecht, and Andrea Vedaldi. *What does CLIP know about a red circle? Visual prompt engineering for VLMs*. 2023. arXiv: 2304.06712 [cs.CV]. URL: https://arxiv.org/abs/2304.06712.

[46] Gemini Team et al. *Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context*. 2024. arXiv: 2403.05530 [cs.CL]. URL: https://arxiv.org/abs/2403.05530.

[47] Gemini Robotics Team et al. *Gemini Robotics: Bringing AI into the Physical World*. 2025. arXiv: 2503.20020 [cs.RO]. URL: https://arxiv.org/abs/2503.20020.

[48] Octo Model Team et al. *Octo: An Open-Source Generalist Robot Policy*. 2024. arXiv: 2405.12213 [cs.RO]. URL: https://arxiv.org/abs/2405.12213.

[49] Anirudh Topiwala, Pranav Inani, and Abhishek Kathpal. *Frontier Based Exploration for Autonomous Robot*. 2018. arXiv: 1806.03581 [cs.RO]. URL: https://arxiv.org/abs/1806.03581.

[50] Claudio Urrea and John Kern. "Recent Advances and Challenges in Industrial Robotics: A Systematic Review of Technological Trends and Emerging Applications". In: *Processes* 13.3 (2025). ISSN: 2227-9717. URL: https://www.mdpi.com/2227-9717/13/3/832.

[51] Beichen Wang et al. *VLM See, Robot Do: Human Demo Video to Robot Action Plan via Vision Language Model*. 2024. arXiv: 2410.08792 [cs.RO]. URL: https://arxiv.org/abs/2410.08792.

[52] Ruoyao Wang et al. *ScienceWorld: Is your Agent Smarter than a 5th Grader?* 2022. arXiv: 2203.07540 [cs.CL]. URL: https://arxiv.org/abs/2203.07540.

[53] Yufei Wang et al. *RL-VLM-F: Reinforcement Learning from Vision Language Foundation Model Feedback*. 2024. arXiv: 2402.03681 [cs.RO]. URL: https://arxiv.org/abs/2402.03681.

[54] Zhijie Wang et al. *Towards Testing and Evaluating Vision-Language-Action Models for Robotic Manipulation: An Empirical Study*. 2024. arXiv: 2409.12894 [cs.SE]. URL: https://arxiv.org/abs/2409.12894.

[55] Jason Wei et al. *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*. 2023. arXiv: 2201.11903 [cs.CL]. URL: https://arxiv.org/abs/2201.11903.

[56] Enze Xie et al. *SegFormer: Simple and Efficient Design for Semantic Segmentation with Transformers*. 2021. arXiv: 2105.15203 [cs.CV]. URL: https://arxiv.org/abs/2105.15203.

[57] Karmesh Yadav et al. *Habitat Challenge 2022*. https://aihabitat.org/challenge/2022/. 2022.

[58] Karmesh Yadav et al. *Habitat-Matterport 3D Semantics Dataset*. 2023. arXiv: 2210.05633 [cs.CV]. URL: https://arxiv.org/abs/2210.05633.

[59] An Yan et al. *GPT-4V in Wonderland: Large Multimodal Models for Zero-Shot Smartphone GUI Navigation*. 2023. arXiv: 2311.07562 [cs.CV]. URL: https://arxiv.org/abs/2311.07562.

[60] Jianwei Yang et al. "Set-of-Mark Prompting Unleashes Extraordinary Visual Grounding in GPT-4V". In: *arXiv preprint arXiv:2310.11441* (2023).

[61] Bangguo Yu, Hamidreza Kasaei, and Ming Cao. "L3mvn: Leveraging large language models for visual target navigation". In: *International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2023.

[62] Jesse Zhang et al. *Bootstrap Your Own Skills: Learning to Solve New Tasks with Large Language Model Guidance*. 2023. arXiv: 2310.10021 [cs.RO]. URL: https://arxiv.org/abs/2310.10021.

[63] Jiazhao Zhang et al. "NaVid: Video-based VLM Plans the Next Step for Vision-and-Language Navigation". In: *arXiv preprint arXiv:2402.15852* (2024).

[64] Gengze Zhou, Yicong Hong, and Qi Wu. *NavGPT: Explicit Reasoning in Vision-and-Language Navigation with Large Language Models*. 2023. arXiv: 2305.16986 [cs.CV]. URL: https://arxiv.org/abs/2305.16986.

[65] Zhiyuan Zhou et al. *Autonomous Improvement of Instruction Following Skills via Foundation Models*. 2024. arXiv: 2407.20635 [cs.RO]. URL: https://arxiv.org/abs/2407.20635.