NavAI: A Generalizable LLM Framework for Navigation Tasks in Virtual Reality Environments

Xue Qin
Department of Computing Sciences
Villanova University
Villanova, U.S.
xue.qin@villanova.edu

Matthew DiGiovanni
Department of Computing Sciences
Villanova University
Villanova, U.S.
mdigio02@villanova.edu

***Abstract*—Navigation is one of the fundamental tasks for automated exploration in Virtual Reality (VR). Existing technologies primarily focus on path optimization in 360-degree image datasets and 3D simulators, which cannot be directly applied to immersive VR environments. To address this gap, we present `NavAI`, a generalizable large language model (LLM)-based navigation framework that supports both basic actions and complex goal-directed tasks across diverse VR applications. We evaluate `NavAI` in three distinct VR environments through goal-oriented and exploratory tasks. Results show that it achieves high accuracy, with an 89% success rate in goal-oriented tasks. Our analysis also highlights current limitations of relying entirely on LLMs, particularly in scenarios that require dynamic goal assessment. Finally, we discuss the limitations observed during the experiments and offer insights for future research directions.**

***Index Terms*—Virtual Reality, Large Language Model, Navigation**

I. Introduction

According to industry forecasts, consumer VR and AR shipments are expected to grow from 17.81 million units in 2023 to 30.88 million by 2026 [1]. With its ability to simulate physical environments, VR is now widely used in applications such as real estate tours, science education, and manufacturing training. To ensure quality at scale, researchers and developers have proposed and applied automated testing [2] in VR applications, focusing on areas such as exploration testing [3], [4], unit testing [5], and privacy enhancement [6].

Among all these automated testing tasks, a successful navigation serves as one of the fundamental starting points for the exploration to help the user interact with the environment effectively. Navigating in a VR space—such as an unknown city or dense forest—can be challenging without guidance, and a map alone is insufficient for addressing spatial navigation, user orientation, and engagement. The current state-of-the-art solution for navigation is called Vision Language Navigation (VLN), which requires the agent to follow human language instructions to navigate in previously unseen environments. Types of VLN include Goal-oriented [7] and Route-oriented [8], [9], and the user may interact with the VLN in single turn or multiple turns [10].

Most discussed VLN techniques have been developed and tested either on real-world environments represented through 360 virtual tours or within traditional 3D simulators. However, these environments differ significantly from fully immersive VR settings. Existing navigation support for VR is typically limited to either pre-defined guides embedded in pre-programmed environments [11] or built-in AI features tailored to specific VR applications [12]. As the VR industry continues to grow, many new and complex environments will be created, and new engagements will be designed. There is a pressing need for robust and scalable navigation solutions that can generalize across both current and emerging VR applications.

In this paper, we propose a large language model (LLM)-based navigation solution framework, named `NavAI`, that can be applied across diverse VR environments. The framework leverages the broad knowledge of LLMs to interpret the virtual world using screenshots captured from the VR application. Specifically, `NavAI` first employs the Comprehensive Interpreter component to analyze the visual scene and generate both visual interpretations and textual descriptions. Then, the Multi-agent Decision Voter parses the user's navigation request and categorizes it into one of the newly defined navigation types. This voter also helps decide whether the intended goal has been achieved. If the navigation task is still in progress, the Decision-to-Control Mapping matches the intent into corresponding control functions and executes navigation actions. `NavAI` currently supports both basic action commands and complex goal-directed navigation tasks.

We evaluate the effectiveness of `NavAI` across three diverse VR environments and assign different navigation tasks of goal-driven and exploratory scenarios. The framework shows consistent performance on basic action tasks such as "move forward" and "turn left", successfully executing all 21 basic motion commands with minimal overhead (average 0.74s). `NavAI` also achieves an 89% success rate (16 out of 18) in direct goal-oriented tasks. However, efficiency varies across environments, with the interpretation component and decision-making voter introducing notable latency, particularly in tasks requiring flexible stopping conditions like exploratory scanning. Finally, we conduct an in-depth discussion on the usability of `NavAI` and address the observed limitations by proposing potential future solutions.

The contributions of this paper are as follows:

- We designed a generalizable framework that can perform

2151-0849/25/$31.00 ©2025 IEEE
DOI 10.1109/ASEW67777.2025.00067

navigation tasks in any VR environment by bridging user requests with interpretations and decisions made by the Large Language Models (LLMs).
- We conduct three sets of experiments covering different task types across diverse VR environments. Results demonstrate that NavAI achieves high success rates and reasonable efficiency across scenarios.
- We analyze the usability of LLM-based navigation, highlighting current limitations and proposing future directions to improve performance and real-time applicability.

II. Framework

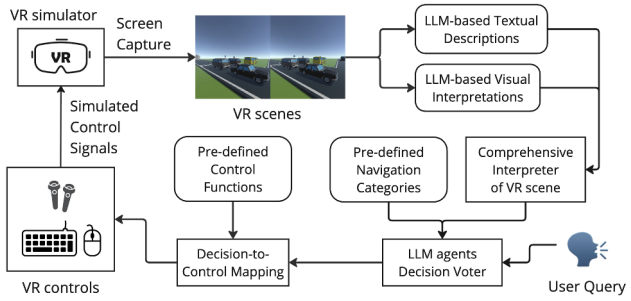

Fig. 1. NavAI Framework overview

In this section, we introduce the NavAI framework, which automatically captures and processes stereoscopic images from a VR application, analyzes the user's query, and then sends the corresponding simulated control signals. As illustrated in Figure 1, the NavAI consists of the following core components for navigation automation: 1) Comprehensive Interpreter – leverages a Large Language Model to generate both textual descriptions and visual interpretations of VR scenes. 2) Predefined Navigation Categories – a dataset covering common navigation types and subtypes to guide user query classification. 3) Decision Voter – aggregates the decisions of multiple LLM agents to categorize the intended navigation action. 4) Decision-to-Control Mapping – matches categorized queries to corresponding control functions and executes them on the VR simulator. Other supporting components, such as the screen capture module and user query receiver, are excluded from this discussion as they are not central to the navigation automation pipeline.

A. Comprehensive Interpreter

Since all the VR scenes are never trained and seen before, the main goal of the comprehensive interpreter is to have the Large Language Model generate a basic understanding of the VR environment so that the user's queries can proceed easily with context. Cognitive theories—particularly dual-coding theory [13]—suggest that the human brain encodes both visual and verbal information when processing an object. Following this perspective, we assign two types of tasks to the LLM when analyzing VR screenshots (representing the user's field of view, FOV): visual interpretation and textual description. In particular, for text description, we adopt the chain-of-thought design from the existing VR study [14] to collect the observed objects' names and their feature descriptions. For visual interpretation, we automatically overlay a grid on top of the screenshots and ask the LLM to provide coordinate information. We do not segment and crop every single object from the screen due to time cost considerations. We also consider coordinates to be more useful, as they can indirectly represent the spatial relationships between object positions and the user's standing or observation point. At the end, we aggregated both types of information as context for the decision voter to decide which navigation action should be performed.

B. Pre-defined Navigation Categories

An existing taxonomy of navigation tasks was built from human instructions in a survey conducted by Wu et al. [15]. The survey examined various navigation tasks in 3D simulators that embed embodied AI, where the built-in AI agents act and interact with the environment through extensive training. In this paper, we focus on navigation tasks that support automated exploration, with particular attention to visiting on-site objects that are visible within the FOV or present in the VR scenes. Figure 2 presents our refined taxonomy, which consists of three major types.

- The first category is the **Semantic Interpreter**, which handles non-navigation queries such as "Where am I?" These tasks can be resolved without sending control signals or triggering movement.
- The second category is the **Action Navigator**, where the user explicitly requests a basic action, such as "move forward" or "turn left." This category is divided into two subtypes: Movement Executor and Stability Executor, as they invoke different groups of control functions.
- The third category is the **Goal Navigator**, where the user specifies a navigation request that links to a target object visible in the FOV, such as "I want to go to the yellow bus." This type is more complex since the user does not specify the particular actions. This task typically requires a sequence of actions and will be completed in multiple turns. We further divide this into two subtypes: Direct Goal Navigator, which refers to goals visible in the current FOV, and Exploratory Goal Navigator, which requires scanning or searching for the target before executing the navigation.

C. Decision Voter

The Decision Voter aggregates the outputs from the Comprehensive Interpreter and the Pre-defined Navigation Categories, and determines the appropriate response for the navigation system based on the user's query. Specifically, it first classifies the query into a navigation category, then evaluates whether the user's requirement has been satisfied or the task goal has been achieved. If the goal is not met, the voter proceeds to the action decision stage to guide the next step.

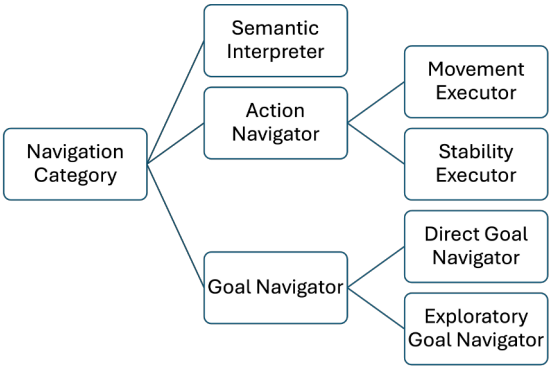

Fig. 2. Navigation Categories for Exploration

Algorithm 1 Decision Voter for Navigation

```
1:  procedure DecisionVoter(query)
2:      ctx ← GetSceneContext()
3:      cat ← ClassifyCategory(query, ctx)
4:      if cat = SEMANTIC_INTERPRETER then
5:          return SEMANTIC_INTERPRETER
6:      else if cat = ACTION_NAVIGATOR then
7:          return ACTION_NAVIGATOR
8:      else
9:          v ← VoteAgents(query, ctx)
10:         if v = REACHED then
11:             StopNav()
12:             return GOAL_REACHED
13:         else
14:             return GOAL_PROGRESS
15:         end if
16:     end if
17: end procedure
```

The detailed algorithm is shown in Algorithm 1. The DecisionVoter procedure receives a user query in natural language and determines the appropriate navigation action. First, the VR scene context is obtained through the method $GetSceneContext()$ at line 2. The query is then classified into one of three categories at line 3. As illustrated from line 4 to line 7, the procedure will return either SEMANTIC_INTERPRETER or ACTION_NAVIGATOR if the category type matches the first two cases. For goal-based navigation, multiple LLM agents vote on whether the intended goal, as referred to in the user query, has been reached, given the scene context. If the goal is confirmed, navigation is stopped and the system returns GOAL_REACHED. Otherwise, the system returns GOAL_PROGRESS, and the following Decision-to-Control Mapping component will decide and execute the next control action.

D. Decision-to-Control Mapping

After determining the goal status, the final Decision-to-Control Mapping component identifies and executes the appropriate control function from a set of pre-defined basic actions. In practice, this mapping connects the user's navigation intent with low-level control commands in the VR environment. We found that OpenAI's function calling feature aligns well with this need, as it allows us to expose the action functions as JSON objects and let the model generate the most suitable invocations, including both parameters and values. This mechanism provides a clean separation between high-level natural language interpretation and concrete execution. The complete set of pre-defined basic actions, categorized into *Movement* and *Stability*, is shown in Table I. Movement actions control positional changes, such as moving forward, while stability actions keep the user in place but adjust orientation or view, including a 360-degree scene scan for comprehensive exploration.

TABLE I
Pre-defined Basic Navigation Actions

Type	Action Functions

Movement	`move_forward(duration)`
	`move_left(duration)`
	`move_right(duration)`

Stability	`in_place_rotate_to_left(duration)`
	`in_place_rotate_to_right(duration)`
	`look_up(duration)`
	`look_down(duration)`
	`scan_360()`

III. Evaluation

In this section, we evaluate the effectiveness of the `NavAI` framework. Specifically, we measure the success rates of different task types across multiple environments and analyze execution times to assess overall usability. We exclude the Semantic Interpreter tasks from evaluation, as they are primarily concerned with the image recognition accuracy of the AI rather than navigation performance.

A. Dataset and Setup

We selected three different Unity VR assets as testing environments. The first is a *Highway* scene, which includes multiple cars and an intersection. The second is a *Ship* scene, featuring an open deck within a narrow space. The third is a *Country House*, an indoor environment containing multiple rooms and objects. To run these assets, we use the Unity Simulator to simulate the VR environments without requiring a physical headset. The simulator is well-suited for functional testing, which aligns with our focus on evaluating the effectiveness of navigation decision algorithms. For the *Comprehensive Interpreter* and the *Decision-to-Signal Mapping* components, we employ GPT-4o [16], following the study by Qi et al. [14]. In the *Decision Voter* component, we integrate three different LLMs for major voting strategies: GPT-4o [16], Grok-2 [17], and Gemini-2.5-Flash [18].

B. Effectiveness of Action Navigator Task

Table II reports the execution durations of `NavAI` when the user's query falls into the Action Navigator category across three VR environments. Each action was designed to last 2

seconds, and all were executed successfully. On average, the measured overhead across all actions and environments is approximately 0.74 seconds. This overhead reflects the combined time cost of GPT-4o in mapping the request to signal functions, the execution of the defined functions within NavAI, and the simulation and rendering costs introduced by the Unity environment. Overall, the results indicate that NavAI can process and execute basic action queries consistently and efficiently across diverse VR contexts.

TABLE II
EXECUTION DURATIONS (IN SECONDS) AND AVERAGE OVERHEAD OF ACTION NAVIGATOR QUERIES ACROSS THREE VR ENVIRONMENTS.

Action	Highway	Country House	Ship	Overhead
Move Forward	2.68	2.74	2.72	0.71
Move Left	2.72	2.62	2.61	0.65
Move Right	2.60	3.49	2.75	1.28
Rotate Left	2.59	2.82	2.63	0.68
Rotate Right	2.45	2.80	2.65	0.63
Look Up	2.76	2.95	2.70	0.80
Look Down	2.60	2.91	2.67	0.73
Overall Average	2.63	2.90	2.67	**0.74**

C. Effectiveness of Direct Goal Navigator Task

We further evaluate the effectiveness of NavAI on Direct Goal-oriented navigation tasks, focusing on both accuracy (success rate) and efficiency (execution time). Representative tasks were tested across three VR environments. In the Highway scene, the task was "*Get to the back of the yellow bus and avoid hitting other cars, going around them.*" In the Country House, the task was "*Walk through the doorway on the left and enter the bedroom.*" In the Ship environment, the task was "*Walk over to the cannon on your right.*" Tables III,IV,V present the execution outcomes across multiple attempts in each environment, along with detailed efficiency measurements in terms of time per navigation turn.

Across all environments, 16 out of 18 attempts were ultimately successful, indicating that NavAI can reliably reach designated goals. However, the efficiency analysis highlights that navigation incurs notable time costs. On average across the three testing environments, the execution time per turn is approximately 49.4 seconds, with an average of 8 turns per attempt. This means that completing a typical goal-oriented navigation task can take up to 395 seconds in total. Within each turn, the Comprehensive Interpreter dominates with about 72% (35.6 seconds), consisting of Visual interpretation (19.8 seconds, 40%) and Textual description (15.8 seconds, 32%). The Decision Voter contributes around 17% (8.3 seconds), while Action Execution accounts for about 9% (4.6 seconds). In summary, the goal-based navigation efficiency is primarily constrained by interpretation and decision-making overhead, and the action execution requires a relatively shorter time.

D. Effectiveness of Exploratory Goal Navigator Task

The 360-degree exploratory scanning task was evaluated across three environments. Table VI shows the three attempts for each environment. Column 3 stands for the number of rotations before a scan was considered complete. On average, the time cost for one completed scan is 41.1 seconds, which is less than one minute. Additionally, each rotation only took 4.6 seconds, which indicates the framework NavAI can finish the task in a timely manner. However, the number of rotations varied widely across environments, ranging from as few as 2 to as many as 12. This variability indicates that the LLM struggled to consistently determine the appropriate stopping condition during exploratory tasks.

IV. DISCUSSION

In this section, we will discuss the usability of adopting the LLM solution on navigation tasks based on the evaluation results of NavAI, and identify the shortcomings that need to be addressed in future work.

A. Usability Discussion

Action Navigator. This component consistently demonstrates high accuracy, correctly analyzing 21 out of 21 user queries' intentions and successfully mapping them to the predefined actions. In addition, the low overhead (approximately 0.72 seconds) shows strong potential in handling the live navigation request. Together, these results highlight Action Navigator's reliable usability and minimal impact on overall performance.

Comprehensive Interpreter. The Comprehensive Interpreter enables accurate perception of the navigation context, supporting the AI model in making reliable decisions. As a result, NavAI successfully completed 16 out of 18 direct goal-oriented navigation tasks. However, this component also introduces substantial latency, which accounts for approximately 72% of the execution time per navigation turn, with visual interpretation contributing 40% and textual description 32%. This indicates that both sub-processes are major bottlenecks and need to be optimized to enhance overall system performance. In summary, while the Comprehensive Interpreter is suitable for time-insensitive tasks such as functional testing, its high latency limits its usability for real-time applications like live support.

Decision Voter. The Decision Voter integrates multiple LLMs to enable robust decision-making, achieving a high success rate of 89% (16 out of 18 attempts). The average overhead is 8.3 seconds, primarily due to sending and waiting for responses from three separate AI models. Therefore, the average time cost per model is 2.76 seconds, which is more acceptable. However, during exploratory scanning tasks, the number of rotations varied widely from 2 to 12, indicating the difficulty that the large language models have in consistently determining the scan termination conditions. This inconsistency reveals the models' limitations in understanding a sequence of image inputs and a potential need for more complicated context management. In summary, while Decision Voter demonstrates its effectiveness in handling the offline testing, its usability decreases in real-time navigation support.

Overall Usability Assessment. Across environments, the NavAI framework demonstrates a high success rate (approx-

TABLE III
EVALUATION RESULTS FOR DIRECT GOAL-ORIENTED NAVIGATION IN THE HIGHWAY ENVIRONMENT.

Attempt	Success	# Turns	Decision Voter (sec.)	Comprehensive Interpreter		Action (sec.)	Total/Turn (sec.)
				Visual (sec.)	Textual (sec.)		
1	Yes	9	8.35	13.02	15.00	3.60	40.51
2	Yes	11	10.33	14.60	16.13	4.52	46.23
3	Yes	10	11.61	13.62	16.18	5.31	47.32
4	Yes	7	13.48	19.78	17.90	3.99	55.88
5	Yes	7	8.26	15.39	16.80	3.82	44.95
6	Yes	13	9.09	17.97	17.92	4.39	50.09
Average	-	**9.5**	**10.19**	**15.73**	**16.66**	**4.27**	**47.50**

TABLE IV
EVALUATION RESULTS FOR DIRECT GOAL-ORIENTED NAVIGATION IN THE COUNTRY HOUSE ENVIRONMENT.

Attempt	Success	# Turns	Decision Voter (sec.)	Comprehensive Interpreter		Action (sec.)	Total/Turn (sec.)
				Visual (sec.)	Textual (sec.)		
1	Yes	5	6.38	17.37	17.32	4.75	46.65
2	Yes	5	5.39	11.23	13.58	3.49	34.17
3	Yes	5	5.53	10.94	15.38	3.08	35.37
4	Yes	5	19.55	17.38	17.38	5.32	63.43
5	Yes	4	7.57	7.12	20.07	7.49	42.59
6	No	19	6.49	14.63	15.92	6.98	44.77
Average	-	**7.2**	**8.82**	**13.11**	**16.61**	**5.18**	**44.50**

TABLE V
EVALUATION RESULTS FOR DIRECT GOAL-ORIENTED NAVIGATION IN THE SHIP ENVIRONMENT.

Attempt	Success	# Turns	Decision Voter (sec.)	Comprehensive Interpreter		Action (sec.)	Total/Turn (sec.)
				Visual (sec.)	Textual (sec.)		
1	Yes	17	7.35	29.53	18.42	5.33	62.00
2	No	4	5.66	20.63	18.31	3.62	49.30
3	Yes	6	6.34	31.51	20.27	3.37	63.14
4	Yes	4	5.18	14.13	15.58	4.47	40.02
5	Yes	7	6.45	27.14	19.30	4.32	58.72
6	Yes	6	6.03	33.92	18.75	4.33	64.40
Average	-	**7.3**	**6.17**	**26.48**	**18.77**	**4.24**	**56.26**

TABLE VI
EVALUATION RESULTS FOR EXPLORATORY SCANNING TASK.

Env.	Att.	Rot.	Time/ Rot. (sec.)	Decision (sec.)	Total/ Att. (sec.)
Highway	1	4	3.65	4.74	19.35
	2	5	3.75	4.92	23.64
	3	5	3.42	3.33	20.44
Country	1	10	5.21	12.67	64.76
	2	8	5.31	5.92	48.42
	3	2	2.87	5.74	14.02
Ship	1	9	4.90	6.66	50.77
	2	10	5.12	8.39	59.57
	3	12	5.32	11.15	75.04

imately 89%) and provides valuable insights into LLM-driven navigation. However, its overall efficiency is constrained by LLM-based interpretation and decision-making costs, and inconsistent termination behavior further reduces scanning task efficiency. In summary, the `NavAI` solution is *reasonable for offline testing and evaluation*, but its usability is *limited for real-time or live support applications* without further optimization.

B. Future Research Insight

To address the usability limitations of the Comprehensive Interpreter, the main goal is to reduce processing time for both text generation and image recognition. Currently, `NavAI` relies entirely on remote LLMs for these tasks, leading to longer delays. A promising direction is to decompose the process into subtasks and develop lightweight local AI models to handle them more efficiently. The key research challenge lies in defining these subtasks and building fast, accurate local models. This aligns with recent trends in edge-based, low-latency LLM design [19], [20].

For the Decision Voter, the main usability issues stem from the sequential communication with multiple LLM agents and the lack of effective context management. The first issue—latency due to sequential processing—can be mitigated by adopting parallel inference and reducing token usage, similar to the strategies proposed in Plurals [21] and LiveMind [22]. The second issue requires designing efficient mechanisms to manage and synchronize context across multiple agents.

V. CONCLUSION

In this paper, we presented NavAI, a large language model-powered framework for automating task-oriented navigation in VR environments. It integrates virtual scene interpretation, navigation query classification, multi-agent decision voting, and control mapping, which simulate navigation actions. NavAI demonstrated strong effectiveness in supporting Action Navigator and Direct Goal-oriented tasks, achieving an 89% success rate across three VR environments. However, the observed overhead from interpretation and decision-making significantly reduces the navigation usability. We conclude with a discussion on these limitations and propose directions for future improvements.

REFERENCES

[1] Fortune Business Insights, "Virtual reality market size," https://www.fortunebusinessinsights.com/industry-reports/virtual-reality-market-101378, 2025, [Online; accessed 2025-07-28].

[2] D. E. Rzig, N. Iqbal, I. Attisano, X. Qin, and F. Hassan, "Virtual reality (vr) automated testing in the wild: A case study on unity-based vr applications," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2023. New York, NY, USA: Association for Computing Machinery, 2023, p. 1269–1281. [Online]. Available: https://doi.org/10.1145/3597926.3598134

[3] X. Wang, "Vrtest: an extensible framework for automatic testing of virtual reality scenes," in *Proceedings of the ACM/IEEE 44th International Conference on Software Engineering: Companion Proceedings*, ser. ICSE '22. New York, NY, USA: Association for Computing Machinery, 2022, p. 232–236. [Online]. Available: https://doi.org/10.1145/3510454.3516870

[4] X. Qin and G. Weaver, "Utilizing generative ai for vr exploration testing: A case study," in *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering Workshops*, ser. ASEW '24. New York, NY, USA: Association for Computing Machinery, 2024, p. 228–232. [Online]. Available: https://doi.org/10.1145/3691621.3694955

[5] A. Gil, T. Figueira, E. Ribeiro, A. Costa, and P. Quiroga, "Automated test of vr applications," in *HCI International 2020 – Late Breaking Posters*, C. Stephanidis, M. Antona, and S. Ntoa, Eds. Cham: Springer International Publishing, 2020, pp. 145–149.

[6] J. Y. Kim, C. Zuo, Y. Zhao, and Z. Lin, "Autovr: Automated ui exploration for detecting sensitive data flow exposures in virtual reality apps," 2025. [Online]. Available: https://arxiv.org/abs/2508.12187

[7] Y. Qi, Q. Wu, P. Anderson, X. Wang, W. Y. Wang, C. Shen, and A. van den Hengel, " REVERIE: Remote Embodied Visual Referring Expression in Real Indoor Environments ," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2020, pp. 9979–9988. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR42600.2020.01000

[8] P. Anderson, Q. Wu, D. Teney, J. Bruce, M. Johnson, N. Sunderhauf, I. Reid, S. Gould, and A. van den Hengel, " Vision-and-Language Navigation: Interpreting Visually-Grounded Navigation Instructions in Real Environments ," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, CA, USA: IEEE Computer Society, Jun. 2018, pp. 3674–3683. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/CVPR.2018.00387

[9] A. Ku, P. Anderson, R. Patel, E. Ie, and J. Baldridge, "Room-across-room: Multilingual vision-and-language navigation with dense spatiotemporal grounding," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, B. Webber, T. Cohn, Y. He, and Y. Liu, Eds. Online: Association for Computational Linguistics, Nov. 2020, pp. 4392–4412. [Online]. Available: https://aclanthology.org/2020.emnlp-main.356/

[10] K. Nguyen, D. Dey, C. Brockett, and B. Dolan, "Vision-based navigation with language-based assistance via imitation learning with indirect intervention," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 12 519–12 529.

[11] J. Collins, K. M. Nicholson, Y. Khadir, A. Stevenson Won, and S. Azenkot, "An ai guide to enhance accessibility of social virtual reality for blind people," in *Proceedings of the 26th International ACM SIGACCESS Conference on Computers and Accessibility*, ser. ASSETS '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3663548.3688498

[12] M. Konenkov, A. Lykov, D. Trinitatova, and D. Tsetserukou, "Vr-gpt: Visual language model for intelligent virtual reality applications," 2024. [Online]. Available: https://arxiv.org/abs/2405.11537

[13] A. Paivio, *Mental Representations: A dual coding approach*. Oxford University Press, 09 1990. [Online]. Available: https://doi.org/10.1093/acprof:oso/9780195066661.001.0001

[14] Z. Qi, H. Li, H. Qin, K. Peng, S. He, and X. Qin, "Harnessing large language model for virtual reality exploration testing: A case study," 2025. [Online]. Available: https://arxiv.org/abs/2501.05625

[15] W. Wu, T. Chang, and X. Li, "Vision-language navigation: A survey and taxonomy," 2022. [Online]. Available: https://arxiv.org/abs/2108.11544

[16] OpenAI, "Gpt-4o technical report," https://openai.com/index/gpt-4o-system-card/, 2024, accessed: 2025-08-18.

[17] xAI, "Introducing grok-2," https://x.ai/blog/grok-2, 2024, accessed: 2025-08-18.

[18] G. DeepMind, "Gemini 2.5 technical overview," https://deepmind.google/models/gemini/flash/, 2025, accessed: 2025-08-18.

[19] D. Pollini, B. V. Guterres, R. S. Guerra, and R. B. Grando, "Reducing latency in llm-based natural language commands processing for robot navigation," 2025. [Online]. Available: https://arxiv.org/abs/2506.00075

[20] Y. Zheng, Y. Chen, B. Qian, X. Shi, Y. Shu, and J. Chen, "A review on edge large language models: Design, execution, and applications," *ACM Comput. Surv.*, vol. 57, no. 8, Mar. 2025. [Online]. Available: https://doi.org/10.1145/3719664

[21] J. Ashkinaze, E. Fry, N. Edara, E. Gilbert, and C. Budak, "Plurals: A system for guiding llms via simulated social ensembles," in *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*, ser. CHI '25. New York, NY, USA: Association for Computing Machinery, 2025. [Online]. Available: https://doi.org/10.1145/3706598.3713675

[22] C. Chen, G. L. Zhang, X. Yin, C. Zhuo, U. Schlichtmann, and B. Li, "Livemind: Low-latency large language models with simultaneous inference," 2024. [Online]. Available: https://arxiv.org/abs/2406.14319