Adaptive Iterative Feedback Prompting for Obstacle-Aware Path Planning via LLMs

Masoud Jafaripour[1], **Shadan Golestan**[1], **Shotaro Miwa**[2], **Yoshihiro Mitsuka**[2], **Osmar R. Zaiane**[1,3]

[1]University of Alberta
[2]Mitsubishi Electric Corporation
[3]Alberta Machine Intelligence Institute

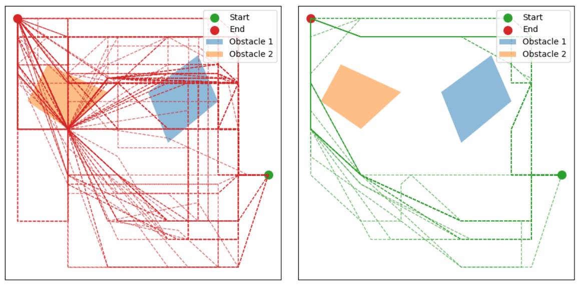
Abstract

Planning is essential for agents operating in complex decision-making tasks, particularly in Human-Robot Interaction (HRI) scenarios, which often require adaptability and the ability to navigate dynamic environments. Large Language Models (LLMs), known for their exceptional natural language understanding capabilities, hold promise for enhancing planning in HRI by processing contextual and linguistic cues. However, their effectiveness is limited by inherent shortcomings in spatial reasoning. Existing LLM-based planning frameworks often depend on combining with classical planning methods or struggle to adapt to dynamic environments, limiting their practical applicability. This paper examines whether the incorporation of an environmental feedback mechanism and iterative planning can enhance the planning capabilities of LLMs. Specifically, we propose the "Adaptive Iterative Feedback Prompting" (**AIFP**) framework for path planning. In **AIFP**, an LLM generates partial trajectories iteratively, which are evaluated for potential collisions using environmental feedback. Based on the evaluation, AIFP executes the trajectory or re-plans. Our preliminary results show that **AIFP** increases the success rate of the baseline by 33.3% and generates efficient, appropriately complex paths, making it a promising approach for dynamic HRI scenarios (project webpage at https://github.com/Masoudjafaripour/AIFP).

Introduction

Planning algorithms provide powerful tools that enable agents to address complex sequential decision-making problems (Eysenbach, Salakhutdinov, and Levine 2019), with applications ranging from robot motion planning (Choset et al. 2005) and solving sliding-tile puzzles (LaValle 2006) to performing high-level tasks such as object manipulation and navigation in kitchens (Brohan et al. 2023). These capabilities are especially critical in Human-Robot Interaction (HRI), where agents must operate in environments that demand collaboration with humans and dynamic adaptation to environment or human intentions (Natarajan et al. 2023).

Path planning and obstacle avoidance are essential for the effective operation of robots, enabling safe and efficient navigation in dynamic environments (Hewawasam, Ibrahim, and Appuhamillage 2022). The literature offers a diverse array of methodologies for solving path planning problem, including classical algorithms such as A* (Hart, Nilsson, and Raphael 1968) and RRT (Kuffner and LaValle 2000), alongside data-driven approaches like reinforcement learning-based search (Zhou, Huang, and Fränti 2022). Despite these advancements, conventional path-planning frameworks struggle with navigating in complex and dynamic environments (Karur et al. 2021). These difficulties are even more pronounced in HRI applications, where robots must not only navigate dynamic environments safely but also adapt to human behavior, preferences, and both physical and language-based instructions (Natarajan et al. 2023).



(a) Unsuccessful paths (b) Successful paths

Figure 1: A 2D path planning task with two obstacles was conducted across 300 trials, generated by the LLM-agent using **naïve few-shot prompting**. Successful and unsuccessful paths are shown separately for clearer illustration.

Large Language Models (LLMs), renowned for their natural language understanding capabilities (Achiam et al. 2023; Touvron et al. 2023), offer promising opportunities for planning in HRI (Zhang and Soh 2023). Their ability to process linguistic and contextual cues makes them well-suited for adaptability and understanding human preferences, particularly in robotic planning (Zhang and Soh 2023; Arora et al. 2024). However, LLM-agents often struggle with effective planning across domains (Liu et al. 2023; Bubeck et al. 2023; Valmeekam et al. 2023; Chen et al. 2024), primarily due to limitations in spatial and long-term temporal reasoning (Aghzal, Plaku, and Yao 2023). For instance, Figure 1 illustrates a 2D navigation task where an agent must navigate to a goal while avoiding obstacles. Using naïve few-shot prompting and repeated 300 times, the paths in Figure 1 reveal GPT-4's inconsistency in generating collision-free paths. However, research shows that integrating LLM-agents into LLM-based frameworks can significantly enhance planning performance (Kambhampati et al. 2024).

An increasing body of research has focused on enhancing the planning capabilities of LLM-agents by integrating environmental feedback, thereby establishing a closed-loop system between the environment and the LLM-agent (Huang et al. 2022; Song et al. 2023; Zhou et al. 2024). This feedback mechanism allows the LLM-agent to interact with the environment, acquire valuable knowledge, and dynamically adjust its planning, resulting in improved performance (Song et al. 2023). For path planning tasks, various hints and feedback derived from ground-truth solutions have been integrated into LLM-planner frameworks, significantly enhancing their effectiveness (Aghzal, Plaku, and Yao 2023; Wu and Mitra 2024).

In this paper, we explore whether integrating real-time environment feedback into an LLM-agent framework through iterative prompting can enhance the spatial reasoning capabilities of LLM-planners in path planning tasks. To achieve this, we propose **AIFP**: the Adaptive Iterative Feedback Prompting framework, which prompts an LLM-agent iteratively to generate partial plans, referred to as segments, for a given path planning task. At each iteration, these segments are evaluated for collisions using the Receding Horizon Planning (RHP) method (Bergman et al. 2020), which predicts the agent's future trajectory over a finite time horizon. If a potential collision is detected, **AIFP** triggers replanning to modify the segment and avoid the collision. Otherwise, part of the segment is executed in the environment. In both cases, feedback, including the current state representation and predicted future states, is provided to the LLM-agent to guide the generation of subsequent segments.

We demonstrate the framework's application through static and dynamic obstacle and goal path planning, a simplified example of robot path planning in HRI scenarios where moving obstacles and goals could represent humans. Preliminary results indicate that LLM-agents using **AIFP** achieve higher success rates in path planning compared to naïve few-shot prompting, across all static environments. These findings encourage expanding the **AIFP** framework to tackle more complex planning scenarios by incorporating rich environmental feedback through Vision-Language Models (VLMs) (Radford et al. 2021; Li et al. 2022) and optimizing trajectories via improved prompting.

Related Work

Few-shot/RAG LLM-agents. Recent studies (Song et al. 2023; Lee, An, and Kim 2024; Xu et al. 2024) demonstrate that the planning abilities of LLMs can be enhanced by leveraging improved in-context learning through enriched contexts, achieved by incorporating few-shot demonstrations (Brown 2020) and employing Retrieval-Augmented Generation (RAG) (Lewis et al. 2020). These methods are effective for high-level tasks, such as identifying steps for cooking, as agents often encounter similar scenarios (e.g., opening a fridge door). However, for low-level tasks, including path planning, LLMs fail to replicate patterns from few-shot demonstrations (Aghzal, Plaku, and Yao 2024). Unlike high-level planning, low-level tasks rely less on abstractions. Instead, our approach, alongside few-shot examples, directly extracts knowledge from the environment to enable grounded actions.

LLM-agent frameworks. Several studies have explored integrating LLMs with planning algorithms to enhance planning capabilities (Brohan et al. 2023; Meng et al. 2024). Ahn et al. (Brohan et al. 2023) proposed the SayCan framework, enabling robots to follow high-level language instructions by using value functions and pre-trained low-level skills to ensure feasible actions. However, its reliance on pr-etrained skills may limit applicability in domains requiring rapid adaptation to new, low-level tasks, especially when such skills are not already learned. Another approach (Meng et al. 2024) uses LLMs to guide an A* algorithm for more effective path planning. Yet, in dynamic HRI environments, this method struggles with computational efficiency due to the complexity of A* cost calculations. In contrast, our approach uses a simple, effective feedback mechanism that requires no training and performs well in dynamic HRI environments.

LLM-agents with feedback. Several works aim to improve the feasibility and accuracy of LLM-generated plans through feedback mechanisms. Recent studies (Sharan et al. 2023; Zhou et al. 2024) introduced novel feedback techniques to handle environmental uncertainties. However, these often require simulating or executing entire trajectories, which can be computationally expensive. Other studies, such as (Singh et al. 2023), ground LLM actions in the current environment state, enabling dynamic plan modifications for high-level, long-horizon tasks. Similarly, **AIFP** integrates feedback by providing LLMs with a representation of the current environment state to ground their actions. Unlike these, **AIFP** uses RHP (Bergman et al. 2020) to evaluate near-future actions at each state and predict potential collisions.

Method

Problem Formulation. In this paper, we focus on path planning in a 2D environment comprising obstacles, an agent, and specified start and goal coordinates. An instance of a

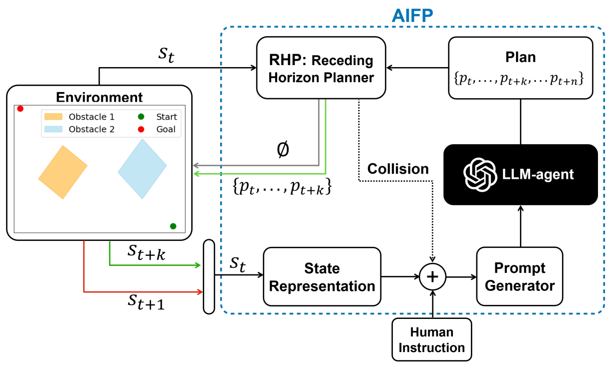


Figure 2: The **AIFP** framework for path planning. Gray and green arrows indicate whether the RHP component detects a collision or not, respectively.

path planning problem is defined by an initial state s_0, a goal state s_g, and a set of obstacles $\mathcal{O} = \{O_1, \dots, O_N\}$ representing prohibited configurations (Choset et al. 2005). At each time-step t, the environment transitions from state s_t to s_{t+1} based on the agent's action p_t. The goal is to determine a sequence of actions $\mathcal{P}$ that transitions the environment from s_0 to s_g while ensuring that the straight line connecting consecutive waypoints avoids all obstacles (Figure 2).

The AIFP Framework

We now present the **AIFP** framework. Figure 2 illustrates its application to our path planning task. The **AIFP** framework comprises five main components, each detailed in the following sections.

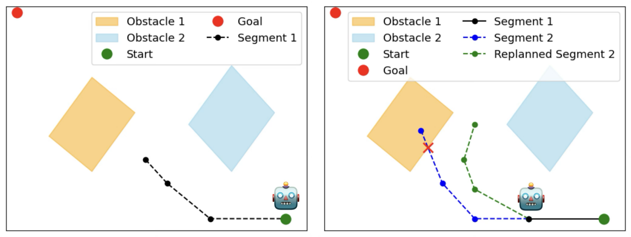
Prompt Generator The initial prompt to the LLM-agent comprises four key components: (1) task description, specifying the task to be performed; (2) scene description, detailing the environment, including the agent's current position, goal, and obstacle locations in textual format; (3) few-shot examples of successful paths; and (4) additional hints and output format (see the Appendix for details). In the case of re-planning, additional environment feedback is appended to the initial prompt, enhancing the agent's environmental perception and instructions for re-planning and exploring alternative routes to generate an improved partial plan.

LLM-agent The LLM-agent plays a central role in generating actions based on the instructions compiled by the Prompt Generator. It receives a prompt as input and generates a sequence of n actions forming a plan, referred to as a *segment* $\mathcal{S}$. This segment is subsequently evaluated to ensure it is collision-free.

Plan A plan refers to a segment comprising a sequence of n actions, i.e., $\mathcal{S}{=}\{p_t, ..., p_{t+k}, ..., p_{t+n}\}$, where each p_i, $i{\in}\{t, ..., t{+}n\}$, represents the agent's (planned) position at time-step i. We denote p_i as a tuple (x_i, y_i), specifying the location of the agent in the environment at time-step i. Each position transition $(p_i{\to}p_{i+1})$ is considered an action, with the step size constrained by predefined bounds: $|\Delta x_i|{\le}\delta$, $|\Delta y_i|{\le}\delta$, and $|\Delta x_i|{>}\epsilon$, $|\Delta y_i|{>}\epsilon$, where δ is the maximum allowed step size, and ϵ is the minimum threshold.

Receding Horizon Planner The RHP component takes the current state of the environment, s_t, and the segment $\mathcal{S}$ as input and evaluates whether executing $\mathcal{S}$ would result in a collision with any obstacle. If no collision is detected, a subset of the segment, comprising the first k actions and denoted as $\mathcal{S}_{t:k}{=}\{p_t, ..., p_{t+k}\}$, is executed in the environment. Otherwise, no action ($\emptyset$) is executed. Figure 3 illustrates two examples of segment evaluation.

State Representation The current location of the agent, goal, and obstacles are embedded in s_t as the current environment state. The State Representation component transforms s_t into a textual description, which is sent to the prompt generator to append to the scene description in the prompt.



(a) Evaluation of a collision free segment with size 3. If k=1, then the agent moves to the next dot in the path.

(b) After the first iteration in (a), RHP detects a collision (blue path), triggering re-planning via the LLM-agent (green path).

Figure 3: Examples of segment evaluation via RHP.

Experiments

We evaluate the impact of iterative feedback prompting on the performance of LLMs in path planning across various static and dynamic environments.

Setup Our **AIFP** framework leverages GPT-4o (Achiam et al. 2023) accessed via the OpenAI API (OpenAI 2023) as the LLM-agent for plan generation. We define a hyper-parameter q as the allowed total number of planning (and re-planning) iterations. To ensure reproducibility, we set the temperature to 0, and limited the outputs to 256 tokens.

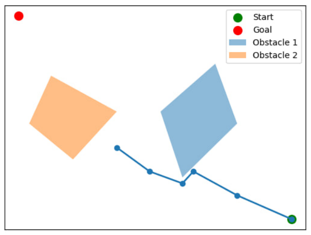
Problems We evaluate the performance of the **AIFP** framework across three path planning problem categories: (1) static and deterministic obstacle configurations, tested over 300 experiments; (2) randomly configured obstacles, where two quadrilateral obstacles' vertices are randomly perturbed by 20% of their initial locations to generate new configurations, with 10 configurations and 20 experiments conducted for each; and (3) either a moving obstacle or a moving goal, both following linear trajectories at speeds of 0.015 and 0.002, respectively, evaluated in 200 experiments.

Metrics Our metric evaluation focuses on assessing the path planning performance of LLMs using three primary metrics which have been used in planning literature (Aghzal, Plaku, and Yao 2023, 2024; Wu and Mitra 2024): (1) Success Rate (**SR%**), which measures the percentage of successful paths that navigate from the start to the goal while satisfying constraints; (2) Planning iterations (**N**), which quantifies the average number of all iterations including feedback loops required to generate a feasible path; and (3) Path length (**PL**), which captures the complexity of solutions by counting the number of points in the final path.

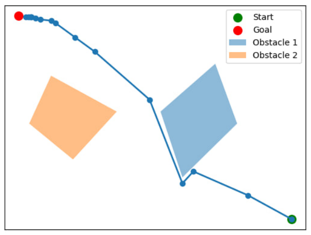
Baseline In experiments with static and randomly positioned obstacles, we compare our method with a naïve few-shot prompting approach (similar to naïve prompting in (Aghzal, Plaku, and Yao 2023)) as a baseline (see Figure 1 and the Appendix for more details) for using LLMs as path planners.

Results

Table 1 presents the results for various categories of path planning problems, comparing the baseline with our **AIFP**



(a) The LLM-agent cannot find a feasible path in q feedback iterations; therefore, it marks this scenario as an unsuccessful plan.



(b) The LLM-agent successfully planned a feasible path from the start to the goal position without any collisions with obstacles.

Figure 4: Examples of planned paths via **AIFP** in an environment with two obstacles.

planning methods. This subsection explores the key observations derived from the table.

Table 1: **AIFP** performance in path planning across different environments

Environment	AIFP			Naïve Prompt		
	SR%	**N**	**PL**	**SR%**	**N**	**PL**
Single Obstacle	55.6	20.6	17.6	22.3	1.0	7.4
Double Obstacles	36.7	24.8	37.7	14.05	1.0	7.6
Random Obstacles	31.5	27.7	29.1	12.5	1.0	6.9
Moving Obstacle	48.5	22.8	24.1	-	-	-
Moving Goal	51.5	14.9	15.4	-	-	-

Note:'-' indicate where results were not applicable for dynamic environments

Naïve few-shot prompting falls short: Naïve few-shot prompting, without feedback and iterative refinement, fails to achieve a high success rate in all static environments. The results of naïve prompt path planning are listed in Table 1. Despite its poor performance in the SR% metric, the non-iterative nature of naïve prompting ensures $N = 1$, indicating low computational requirements. Additionally, the path lengths in all naïve prompting experiments are substantially shorter than those in **AIFP** experiments indicating simplicity of generated plans by LLM-agent in this scenario.

Static obstacle avoidance of AIFP is promising: Our proposed **AIFP** method outperforms the naïve prompting baseline in path planning success rate within static environments, including single, double, and random obstacles (see Table 1), achieving more than twice the success rate. However, it requires significantly more computation, as the number of iterations in **AIFP** is at least 20 times greater than that of naïve prompting. In Figure 4, we present results of the **AIFP** framework in an environment with two obstacles.

AIFP planning is robust to moving obstacle and goal: Experimental results (Table 1) show that **AIFP** can adaptively solve path planning in environments with one moving obstacle or goal, albeit with a slightly lower SR% than in static environments. The success rate for a single moving obstacle is lower than that for a moving target, and planning requires more iterations and a more complex path to solve.

Discussion and Future Work

The experimental results highlight the strengths and tradeoffs of the **AIFP** framework in diverse path planning scenarios. Compared to naïve few-shot prompting, **AIFP**, leveraging an iterative prompting mechanism, achieves higher success rates and can generate more complex path geometries. However, it requires significantly more iterations (over 20 times) than the naïve approach. Despite higher computational cost, using pre-trained LLMs (solely for inference) for path planning with **AIFP** eliminates the need for a training phase, as required in data-driven or learning-based methods.

The results demonstrate **AIFP**'s ability to adapt generated plans to moving obstacles and goal, which is achieved by implementing a receding horizon planning approach. This approach grounds segments by ensuring the prediction horizon (segment size, n) is always longer than the implementation horizon (k, see Figure 2). This allows AIFP to plan further into the future while executing only a small portion of the planned path conservatively, accounting for the possibility of environmental changes.

One main limitation of our approach is that the planned paths are not optimal (see Figure 4), as they are generated by an LLM-agent over a short horizon (each segment). This can be partially addressed by exploring more routes (e.g., using graph representations (Shang, Zhu, and Huang 2024)) and adding evaluation heuristics to the prompt to select more optimal routes. Another limitation for real-world applications is that a global map is provided, while some mobile robot applications rely only on local observations, requiring autonomous mapping. This could be addressed by adding a memory of observations to enrich the LLM context and gradually build a global map.

Future studies should explore the potential of **AIFP** in real-world HRI tasks with human collaborators. Additionally, verbalized feedback could further help LLM-agents better understand human intentions, aligning their actions with expectations. To enhance applicability in HRI, incorporating explainability techniques for LLM-agents (Malhi, Knapic, and Främling 2020; Papagni et al. 2023; Taghian et al. 2024) will improve transparency and interpretability.

Conclusion

This study investigates the use of the adaptive iterative feedback prompting (**AIFP**) framework to leverage GPT-4 for solving 2D path planning problems. The **AIFP** method shows significant improvement over naïve few-shot prompting techniques in static scenarios, highlighting the effectiveness of iterative prompting. However, its performance declines slightly in more complex settings, such as random obstacles or dynamic environments. Random obstacle perturbations reveal further challenges, including increased iterations and more complext paths. While **AIFP** demonstrates promising adaptability, limitations persist, such as suboptimal path generation and the reliance on a global map. Future work will address these issues by incorporating heuristics, graph-based path representations, and memory mechanisms to construct a global map, as well as exploring applications in real-world environments.

References

Achiam, J.; Adler, S.; Agarwal, S.; Ahmad, L.; Akkaya, I.; Aleman, F. L.; Almeida, D.; Altenschmidt, J.; Altman, S.; Anadkat, S.; et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Aghzal, M.; Plaku, E.; and Yao, Z. 2023. Can large language models be good path planners? a benchmark and investigation on spatial-temporal reasoning. *arXiv preprint arXiv:2310.03249*.

Aghzal, M.; Plaku, E.; and Yao, Z. 2024. Look Further Ahead: Testing the Limits of GPT-4 in Path Planning. *arXiv preprint arXiv:2406.12000*.

Arora, R.; Singh, S.; Swaminathan, K.; Datta, A.; Banerjee, S.; Bhowmick, B.; Jatavallabhula, K. M.; Sridharan, M.; and Krishna, M. 2024. Anticipate & Act: Integrating LLMs and Classical Planning for Efficient Task Execution in Household Environments. In *International Conference on Robotics and Automation*.

Bergman, K.; Ljungqvist, O.; Glad, T.; and Axehill, D. 2020. An optimization-based receding horizon trajectory planning algorithm. *IFAC-PapersOnLine*, 53(2): 15550–15557.

Brohan, A.; Chebotar, Y.; Finn, C.; Hausman, K.; Herzog, A.; Ho, D.; Ibarz, J.; Irpan, A.; Jang, E.; Julian, R.; et al. 2023. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on robot learning*, 287–318. PMLR.

Brown, T. B. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Bubeck, S.; Chandrasekaran, V.; Eldan, R.; Gehrke, J.; Horvitz, E.; Kamar, E.; Lee, P.; Lee, Y. T.; Li, Y.; Lundberg, S.; et al. 2023. Sparks of artificial general intelligence: Early experiments with gpt-4. *arXiv preprint arXiv:2303.12712*.

Chen, Y.; Arkin, J.; Dawson, C.; Zhang, Y.; Roy, N.; and Fan, C. 2024. Autotamp: Autoregressive task and motion planning with llms as translators and checkers. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 6695–6702. IEEE.

Choset, H.; Lynch, K. M.; Hutchinson, S.; Kantor, G. A.; and Burgard, W. 2005. *Principles of robot motion: theory, algorithms, and implementations*. MIT press.

Eysenbach, B.; Salakhutdinov, R. R.; and Levine, S. 2019. Search on the Replay Buffer: Bridging Planning and Reinforcement Learning. In Wallach, H.; Larochelle, H.; Beygelzimer, A.; d'Alché-Buc, F.; Fox, E.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics*, 4(2): 100–107.

Hewawasam, H.; Ibrahim, M. Y.; and Appuhamillage, G. K. 2022. Past, present and future of path-planning algorithms for mobile robot navigation in dynamic environments. *IEEE Open Journal of the Industrial Electronics Society*, 3: 353–365.

Huang, W.; Xia, F.; Xiao, T.; Chan, H.; Liang, J.; Florence, P.; Zeng, A.; Tompson, J.; Mordatch, I.; Chebotar, Y.; et al. 2022. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*.

Kambhampati, S.; Valmeekam, K.; Guan, L.; Verma, M.; Stechly, K.; Bhambri, S.; Saldyt, L. P.; and Murthy, A. B. 2024. Position: LLMs Can't Plan, But Can Help Planning in LLM-Modulo Frameworks. In *Forty-first International Conference on Machine Learning*.

Karur, K.; Sharma, N.; Dharmatti, C.; and Siegel, J. E. 2021. A survey of path planning algorithms for mobile robots. *Vehicles*, 3(3): 448–468.

Kuffner, J. J.; and LaValle, S. M. 2000. RRT-connect: An efficient approach to single-query path planning. In *Proceedings 2000 ICRA. Millennium Conference. IEEE International Conference on Robotics and Automation. Symposia Proceedings (Cat. No. 00CH37065)*, volume 2, 995–1001. IEEE.

LaValle, S. M. 2006. *Planning algorithms*. Cambridge university press.

Lee, M.; An, S.; and Kim, M.-S. 2024. PlanRAG: A Plan-then-Retrieval Augmented Generation for Generative Large Language Models as Decision Makers. *arXiv preprint arXiv:2406.12430*.

Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Küttler, H.; Lewis, M.; Yih, W.-t.; Rocktäschel, T.; et al. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33: 9459–9474.

Li, L. H.; Zhang, P.; Zhang, H.; Yang, J.; Li, C.; Zhong, Y.; Wang, L.; Yuan, L.; Zhang, L.; Hwang, J.-N.; et al. 2022. Grounded language-image pre-training. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 10965–10975.

Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; and Stone, P. 2023. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

Malhi, A.; Knapic, S.; and Främling, K. 2020. Explainable agents for less bias in human-agent decision making. In *Explainable, Transparent Autonomous Agents and Multi-Agent Systems: Second International Workshop, EXTRAAMAS 2020, Auckland, New Zealand, May 9–13, 2020, Revised Selected Papers 2*, 129–146. Springer.

Meng, S.; Wang, Y.; Yang, C.-F.; Peng, N.; and Chang, K.-W. 2024. Llm-a*: Large language model enhanced incremental heuristic search on path planning. *arXiv preprint arXiv:2407.02511*.

Natarajan, M.; Seraj, E.; Altundas, B.; Paleja, R.; Ye, S.; Chen, L.; Jensen, R.; Chang, K. C.; and Gombolay, M. 2023. Human-robot teaming: grand challenges. *Current Robotics Reports*, 4(3): 81–100.

OpenAI. 2023. OpenAI API. https://platform.openai.com.

Papagni, G.; de Pagter, J.; Zafari, S.; Filzmoser, M.; and Koeszegi, S. T. 2023. Artificial agents' explainability to support trust: considerations on timing and context. *Ai & Society*, 38(2): 947–960.

Radford, A.; Kim, J. W.; Hallacy, C.; Ramesh, A.; Goh, G.; Agarwal, S.; Sastry, G.; Askell, A.; Mishkin, P.; Clark, J.; et al. 2021. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, 8748–8763. PMLR.

Shang, W.; Zhu, X.; and Huang, X. 2024. Path-LLM: A Shortest-Path-based LLM Learning for Unified Graph Representation. *arXiv preprint arXiv:2408.05456*.

Sharan, S.; Pittaluga, F.; Chandraker, M.; et al. 2023. Llm-assist: Enhancing closed-loop planning with language-based reasoning. *arXiv preprint arXiv:2401.00125*.

Singh, I.; Blukis, V.; Mousavian, A.; Goyal, A.; Xu, D.; Tremblay, J.; Fox, D.; Thomason, J.; and Garg, A. 2023. Progprompt: Generating situated robot task plans using large language models. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, 11523–11530. IEEE.

Song, C. H.; Wu, J.; Washington, C.; Sadler, B. M.; Chao, W.-L.; and Su, Y. 2023. Llm-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2998–3009.

Taghian, M.; Miwa, S.; Mitsuka, Y.; Günther, J.; Golestan, S.; and Zaiane, O. 2024. Explainability of deep reinforcement learning algorithms in robotic domains by using Layer-wise Relevance Propagation. *Engineering Applications of Artificial Intelligence*, 137: 109131.

Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Valmeekam, K.; Marquez, M.; Sreedharan, S.; and Kambhampati, S. 2023. On the planning abilities of large language models-a critical investigation. *Advances in Neural Information Processing Systems*, 36: 75993–76005.

Wu, E.; and Mitra, S. 2024. Can LLMs plan paths with extra hints from solvers? *arXiv preprint arXiv:2410.05045*.

Xu, W.; Wang, M.; Zhou, W.; and Li, H. 2024. P-RAG: Progressive Retrieval Augmented Generation For Planning on Embodied Everyday Task. In *Proceedings of the 32nd ACM International Conference on Multimedia*, 6969–6978.

Zhang, B.; and Soh, H. 2023. Large language models as zero-shot human models for human-robot interaction. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 7961–7968. IEEE.

Zhou, C.; Huang, B.; and Fränti, P. 2022. A review of motion planning algorithms for intelligent robots. *Journal of Intelligent Manufacturing*, 33(2): 387–424.

Zhou, Z.; Song, J.; Yao, K.; Shu, Z.; and Ma, L. 2024. Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, 2081–2088. IEEE.

Appendix

We present below a detailed description of the prompt used for the **naïve few-shot prompting** method. This process was run 300 times, and the resulting paths are plotted in Fig. 1.

Prompt:

Task Description

You are tasked with planning a path based on specific instructions in a 2D environment containing objects and an obstacle. The path must navigate from a starting position to the final goal. Avoid entering the obstacles (blue and orange boxes) at any point.

Environment Description

Environment Details:

- **Start:** [name: start point, shape: circle, color: green, center position: [-1.5, 1.2]]
- **Goal:** [name: goal point, shape: circle, color: red, center position: [1.0, -0.5]]
- **Obstacle 1:** [name: Obstacle-1, shape: quadrilateral, color: blue, corners at [[0.0, -0.15], [0.5, 0.3], [0.3, 0.8], [-0.2, 0.4]]]
- **Obstacle 2:** [name: Obstacle-2, shape: quadrilateral, color: orange, corners at [[-1.0, 0.0], [-0.6, 0.4], [-1.2, 0.7], [-1.4, 0.3]]]

Important: The blue and orange areas are restricted zones and must be avoided completely during the path. No point in the path can intersect or enter this area.

Detail Instructions

Navigation Instructions:

- Start at the green box, then navigate in 2D plane and finally end at the red box.
- Avoid entering the obstacle areas (blue and orange) at any point during the path.
- The path should include curved motions or additional waypoints to smoothly navigate around obstacles and restricted zones.

Obstacle Avoidance Instruction:

1. Identify which corners of the obstacle areas are at the lower side (downside) and which ones are at the upper side (upside) based on their y-coordinates. Assume that a larger y-coordinate indicates the upper direction.
2. Identify which corners of the obstacle areas are on the left side and which ones are on the right side based on their x-coordinates. Assume that a larger x-coordinate indicates the right direction.
3. Assign the following labels to the corners of the obstacle area:
   - Left top corner: C1
   - Right top corner: C2
   - Right bottom corner: C3
   - Left bottom corner: C4

4. For each corner of the obstacle areas, the path must avoid specific directions based on the corner's position. Use the following rules for path planning:
   - For C1 (left top corner): The path must stay either on its left, on its top, or both on its left and top.
     - The path (waypoints of path) must **never** be directly in the region below and the right of C1.
     - However, the path can move above and right of C1, or below and left of C1.
   - For C2 (right top corner): The path must stay either on its right, on its top, or both on its right and top.
     - The path (waypoints of path) must **never** be directly in the region below and the left of C2.
     - However, the path can move above and left of C2, or below and right of C2.
   - For C3 (right bottom corner): The path must stay either on its right, on its bottom, or both on its right and bottom.
     - The path (waypoints of path) must **never** be directly in the region above and the left of C3.
     - However, the path can move below and right of C3, or above and left of C3.
   - For C4 (left bottom corner): The path must stay either on its left, on its bottom, or both on its left and bottom.
     - The path (waypoints of path) must **never** be directly in the region above and the right of C4.
     - However, the path can move below and left of C4, or above and right of C4.
5. Ensure that no path points are placed on the edges of the obstacle areas. All path points should be located entirely outside the obstacle areas, avoiding any points along its boundaries.

Few-Shot Examples

Examples for Reference:

- **Example 1:** Starting from the green cirle and ending at the red circle while avoiding the obstacle area can be done by this trajectory or path:

  ```
  {[1,-0.5],[0.0,-0.25],[-0.35,0.5],
  [-0.8,0.8],[-1.5,1.2]}
  ```

- **Example 2:** Starting from the green circle and ending at the red circle while avoiding the obstacle area can be done by this trajectory or path:

  ```
  {[1,-0.5],[0.0,-0.35],[-1,-0.25],
  [-1.5,0.25],[-1.5,1.2]}
  ```

Output Format

Output Format: Please generate the trajectory as a list of coordinates in the format [x, y], structured as a JSON object like this:

```
{   "Trajectory": [
        [x1, y1],
        [x2, y2],
                ...]}
```

Additional Notes:

- Ensure the JSON is properly formatted with correct syntax.
- Include curved motions where necessary to avoid the osbtacles area.
- The path must consider the entire area of the obstacles, avoiding it completely.

GPT-4 Output:

Then, executing this prompt, GPT-4 will generate an output similar to the box below which is a series of point from start to goal:

Output Format

LLM Output: "Trajectory": [[1.0, -0.5], [0.5, -0.5], [0.5, 0.0], [0.7, 0.5], [0.7, 1.0], [0.5, 1.2], [0.0, 1.2], [-1.0, 1.2], [-1.5, 1.2]]